ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
Τμήμα Πληροφορικής

Εργασία Μαθήματος «Προγραμματισμός στο διαδίκτυο και στον παγκόσμιο ιστό»

ΤΕΛΙΚΟ ΠΑΡΑΔΟΤΕΟ ΜΑΘΗΜΑΤΟΣ	*Δημιουργία 3-tier εφαρμογής για την διαχείριση ραντεβού ιατρικών εξετάσεων*
Όνομα φοιτητή – Αρ. Μητρώου (όλων σε περίπτωση ομαδικής εργασίας)	ΠΑΝΑΓΙΩΤΟΠΟΥΛΟΣ ΔΗΜΗΤΡΙΟΣ Π19130
	ΑΥΓΕΡΙΝΟΣ ΧΡΗΣΤΟΣ Π19020
	ΑΘΑΝΑΣΙΟΣ ΒΙΤΑΚΗΣ Π19247
Ημερομηνία παράδοσης	13/7/2021

Εκφώνηση της άσκησης

Στην τελική εργασία του μαθήματος θα επεκτείνετε την 2η Άσκηση ώστε να ολοκληρώσετε την εφαρμογή
τριών επιπέδων (3-tier), η οποία θα υλοποιεί όλες τις λειτουργίες (μεθόδους) που ορίσατε στην 1η Άσκηση
(με τις πιθανές αλλαγές που έγιναν).

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

1 Γενική Περιγραφή της λύσης

Σε αυτή την άσκηση επεκτείναμε και αναβαθμίσαμε την λειτουργία των Servlets της προηγούμενης εργασίας με την τεχνολογία MVC, προσθέσαμε κρυπτογράφηση των password

Και υλοποιήσαμε τις υπόλοιπες λειτουργίες των Ασθενών, Ιατρών και Διαχειριστών.

2 Κεντρικό μενού

Home: *Κύρια Σελίδα*

Patient: *Σελίδα σύνδεσης ασθενή*

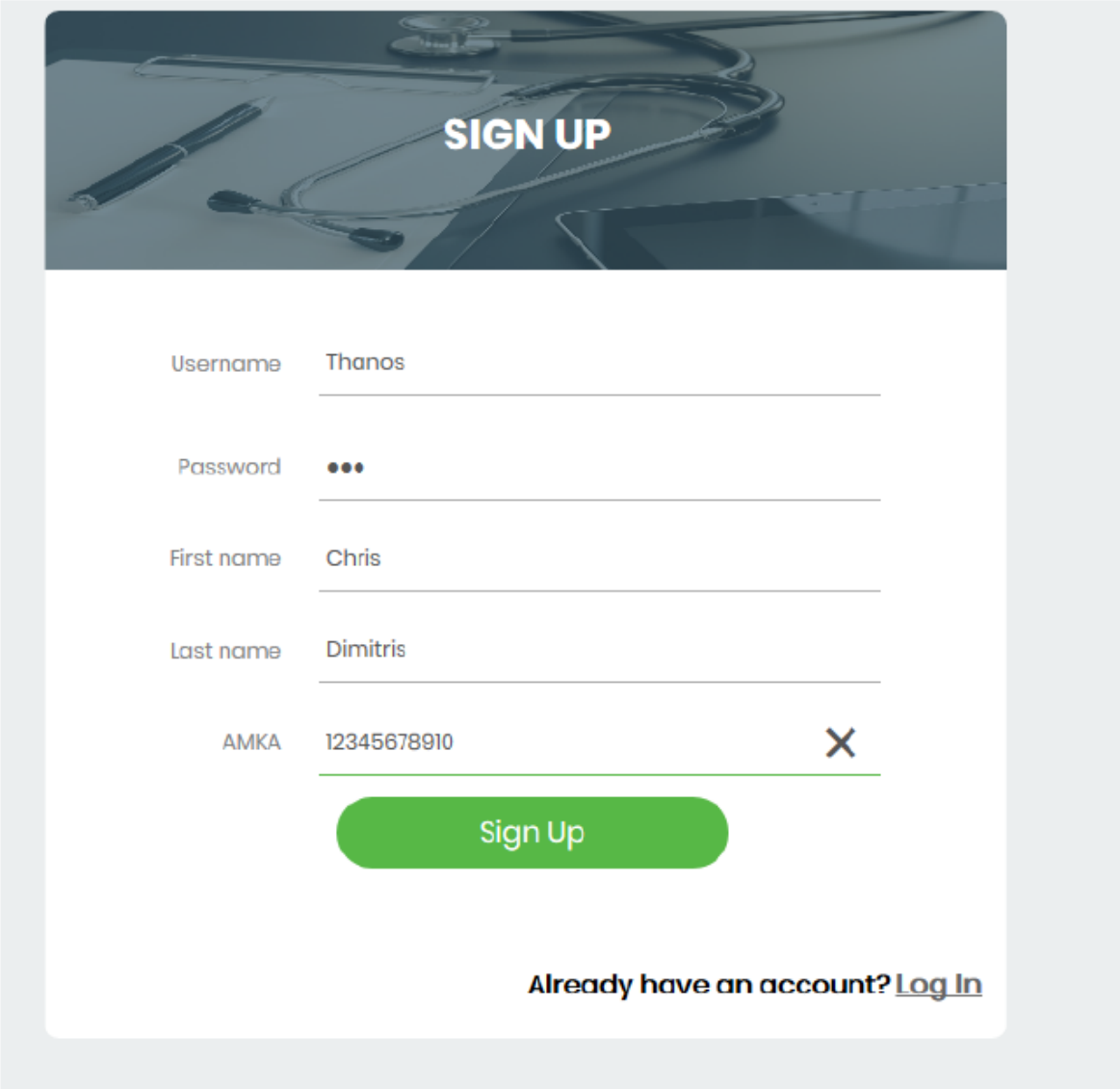
Doctor: *Σελίδα σύνδεσης γιατρού*

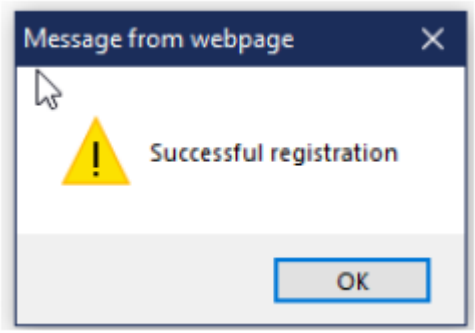
Admin: *Σελίδα σύνδεσης Admin*

About: *Πληροφορίες για τους δημιουργούς της εφαρμογής*

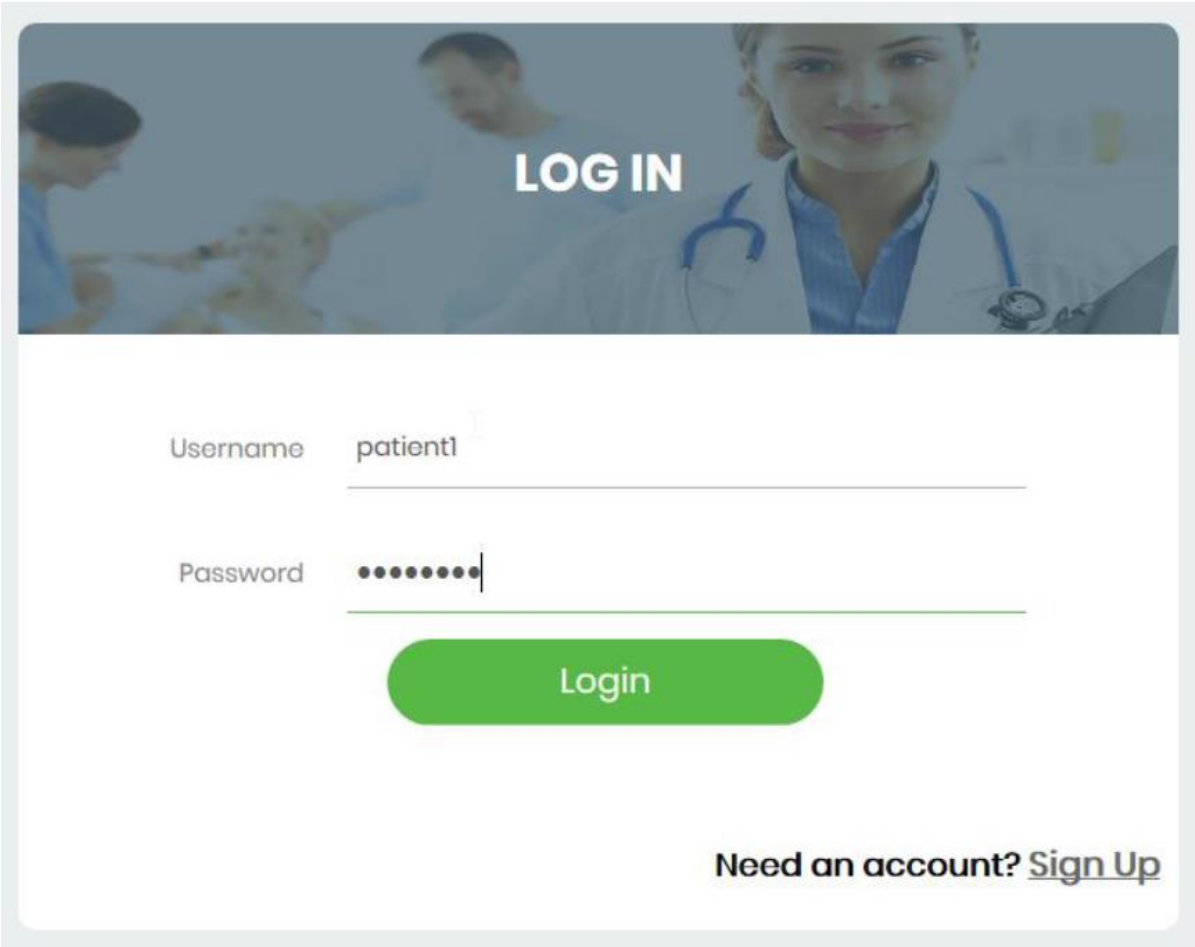
3 Λειτουργίες Ασθενών (Patient):

3.1 Λειτουργία εγγραφής ασθενή





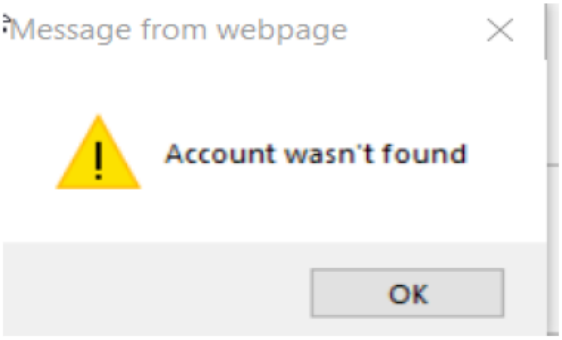
3.2 Λειτουργία σύνδεσης (login) για τον Ασθενή



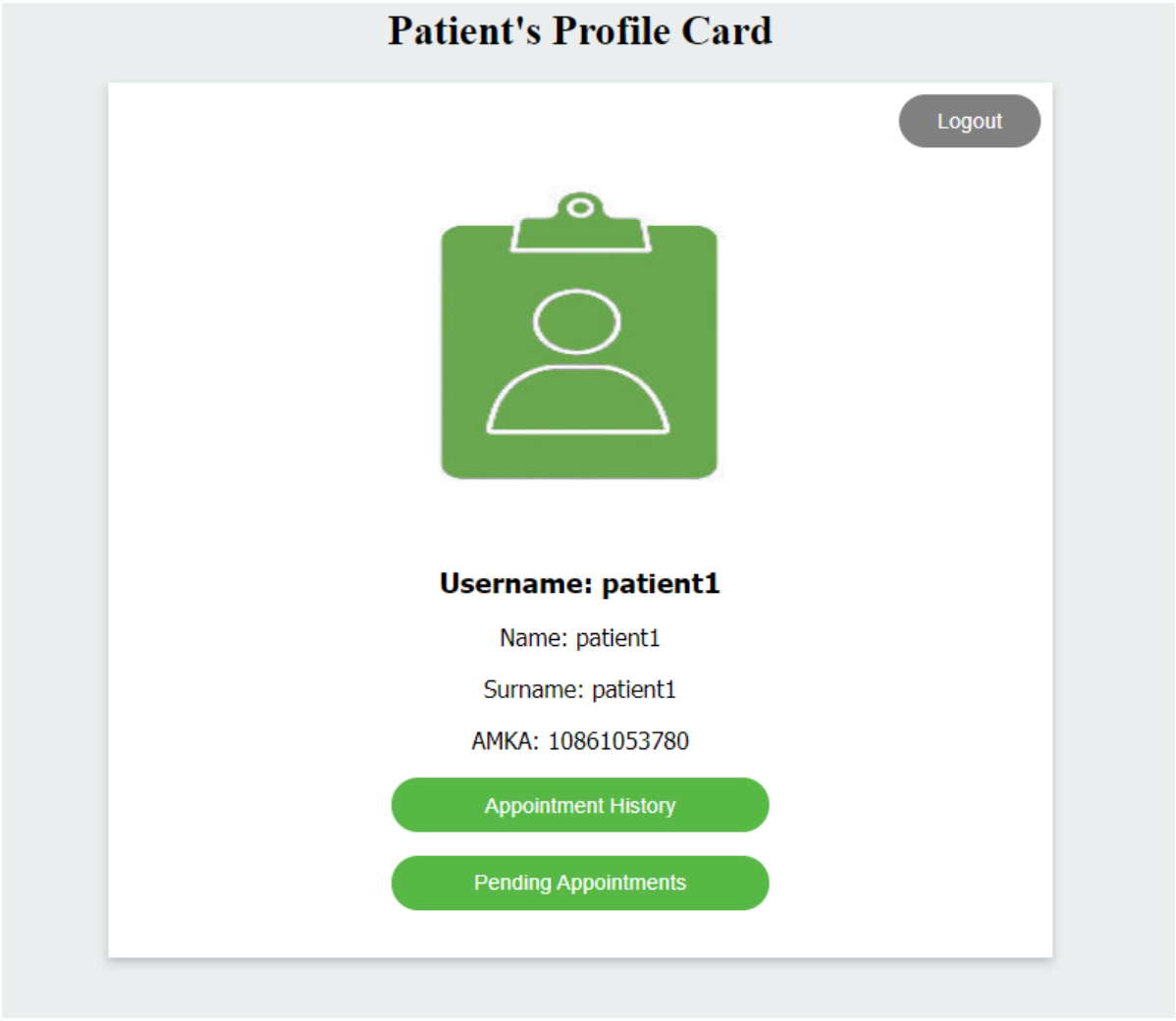
Username: patient1

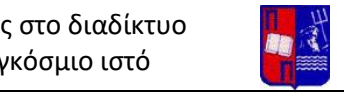
Password: patient1

Σε περίπτωση που δωθούν λάθος στοιχεία

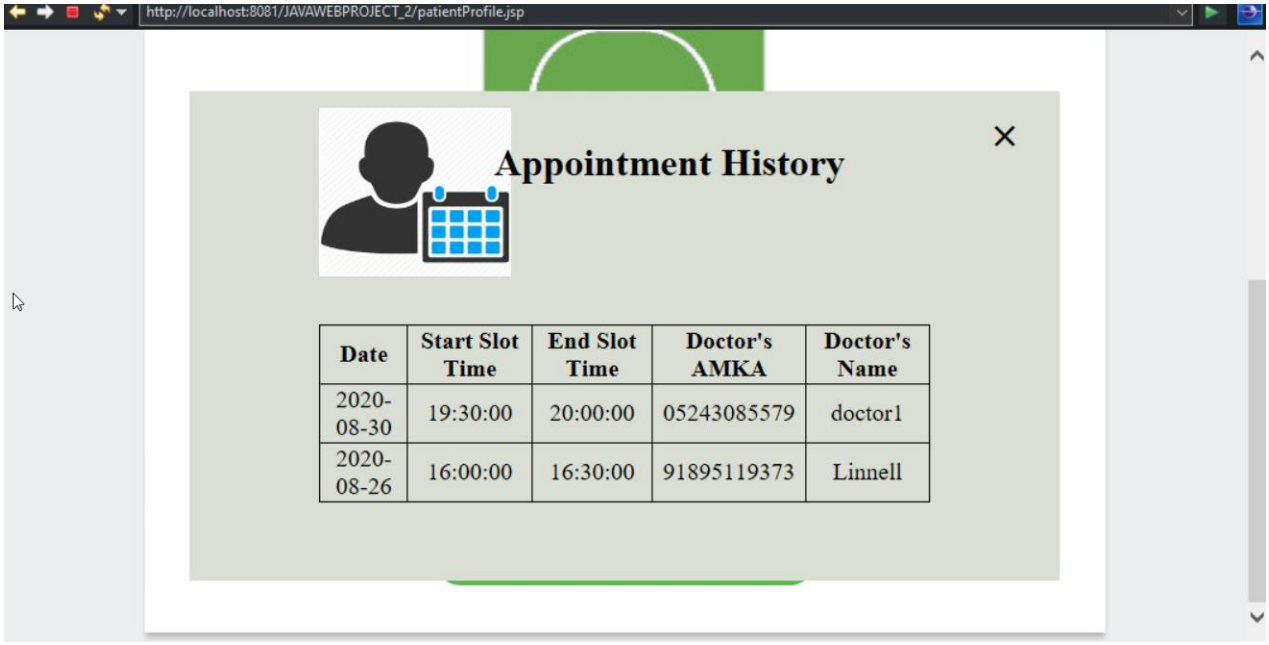


3.3 Προβολή στοιχείων του ασθενή (μόνο τα στοιχεία του ασθενούς που έχει συνδεθεί).





3.4 Προβολή ιστορικού προηγούμενων ραντεβού.



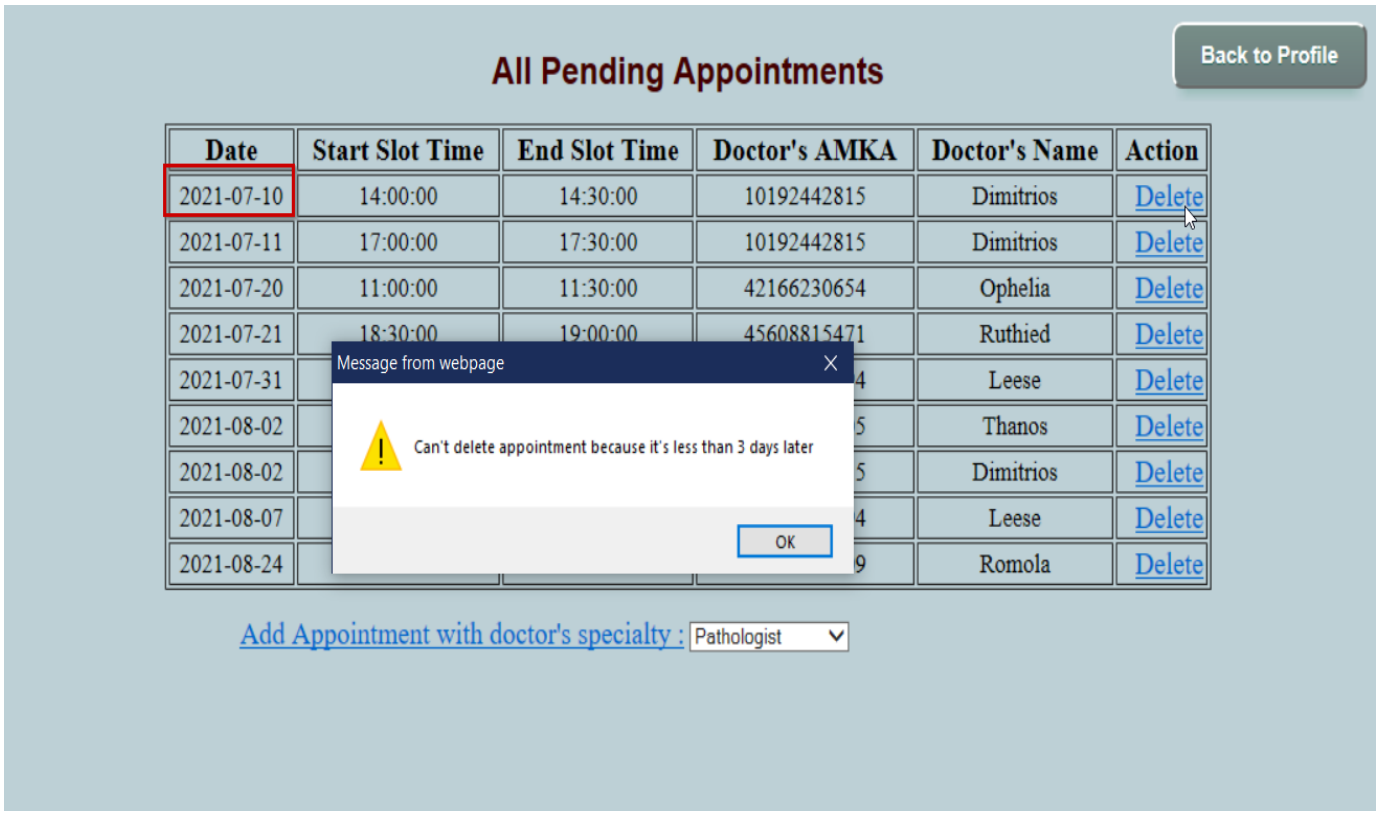
Date	Start Slot Time	End Slot Time	Doctor's AMKA	Doctor's Name
2020-08-30	19:30:00	20:00:00	05243085579	doctor1
2020-08-26	16:00:00	16:30:00	91895119373	Linnell

3.5 Προβολή των προσεχών ραντεβού του συγκεκριμένου ασθενή.

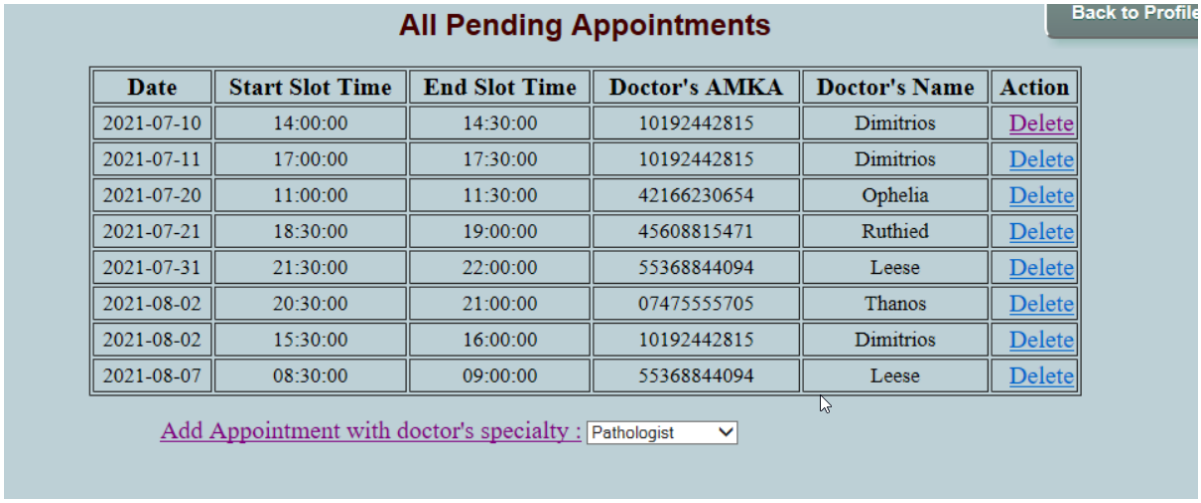
Date	Start Slot Time	End Slot Time	Doctor's AMKA	Doctor's Name	Action
2021-07-14	10:00:00	10:30:00	10192442815	Dimitrios	Delete
2021-07-20	14:00:00	14:30:00	41381050148	Brenn	Delete
2021-07-21	16:00:00	16:30:00	10192442815	Dimitrios	Delete
2021-07-22	17:30:00	18:00:00	10192442815	Dimitrios	Delete
2021-07-26	13:30:00	14:00:00	55670497446	Roger	Delete
2021-07-29	20:00:00	20:30:00	05243085579	doctor1	Delete
2021-08-04	09:00:00	09:30:00	07475555705	Thanos	Delete
2021-08-17	09:00:00	09:30:00	07475555705	Thanos	Delete
2021-08-21	09:30:00	10:00:00	98821388042	Jeffy	Delete

Add Appointment with doctor's specialty : Pathologist

3.5.1 Λειτουργία ακύρωσης προσεχώς ραντεβού



Περίπτωση που κάποιος πάει να διαγράψει ραντεβού που δεν είναι τουλάχιστον 3 ημέρες μετά.



Κανονική διαγραφή(τελευταία εγγραφή).

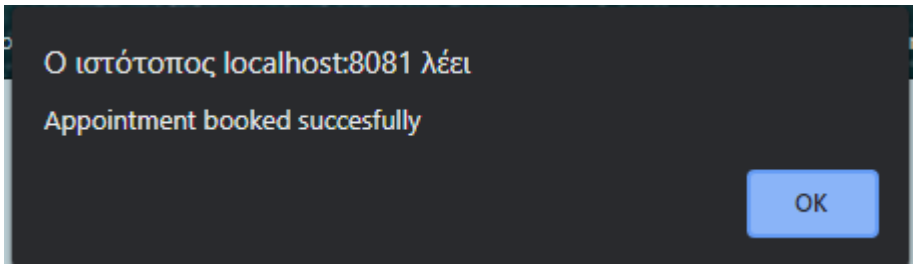
3.5.2 Λειτουργία Κλείσιμου /προβολής διαθέσιμων ραντεβού με γιατρό ειδικότητας

2021-07-20	11:00:00	11:30:00	42166230654	Ophelia	Delete
2021-07-21	18:30:00	19:00:00	45608815471	Ruthied	Delete
2021-07-31	21:30:00	22:00:00	55368844094	Leese	Delete
2021-08-02	20:30:00	21:00:00	07475555705	Thanos	Delete
2021-08-02	15:30:00	16:00:00	10192442815	Dimitrios	Delete
2021-08-07	08:30:00	09:00:00	55368844094	Leese	Delete

Add Appointment with doctor's specialty : Pathologist
Ophthalmologist
Orthopedic

Add Appointment

Date	Start Slot Time	End Slot Time	Doctor's AMKA	Doctor's Name	Action
2021-07-14	10:00:00	10:30:00	10192442815	Dimitrios	Book Appointment
2021-07-16	15:30:00	16:00:00	76682476573	Mariellen	Book Appointment
2021-07-20	14:00:00	14:30:00	41381050148	Brenn	Book Appointment
2021-07-20	19:00:00	19:30:00	41381050148	Brenn	Book Appointment
2021-07-21	16:00:00	16:30:00	10192442815	Dimitrios	Book Appointment
2021-07-21	20:30:00	21:00:00	76682476573	Mariellen	Book Appointment
2021-07-22	17:30:00	18:00:00	10192442815	Dimitrios	Book Appointment
2021-08-03	15:30:00	16:00:00	41381050148	Brenn	Book Appointment
2021-08-03	13:00:00	13:30:00	68968421329	Stace	Book Appointment
2021-08-07	18:30:00	19:00:00	45608815471	Ruthied	Book Appointment
2021-08-13	08:00:00	08:30:00	45608815471	Ruthied	Book Appointment
2021-08-13	10:30:00	11:00:00	96601370317	Redford	Book Appointment
2021-08-15	17:30:00	18:00:00	10192442815	Dimitrios	Book Appointment
2021-08-16	12:30:00	13:00:00	68968421329	Stace	Book Appointment
2021-08-18	14:30:00	15:00:00	68968421329	Stace	Book Appointment
2021-08-21	16:00:00	16:30:00	45608815471	Ruthied	Book Appointment
2021-08-27	19:00:00	19:30:00	45608815471	Ruthied	Book Appointment
2021-08-27	18:30:00	19:00:00	96601370317	Redford	Book Appointment
2021-08-29	09:00:00	09:30:00	76682476573	Mariellen	Book Appointment
2021-08-30	16:00:00	16:30:00	68968421329	Stace	Book Appointment



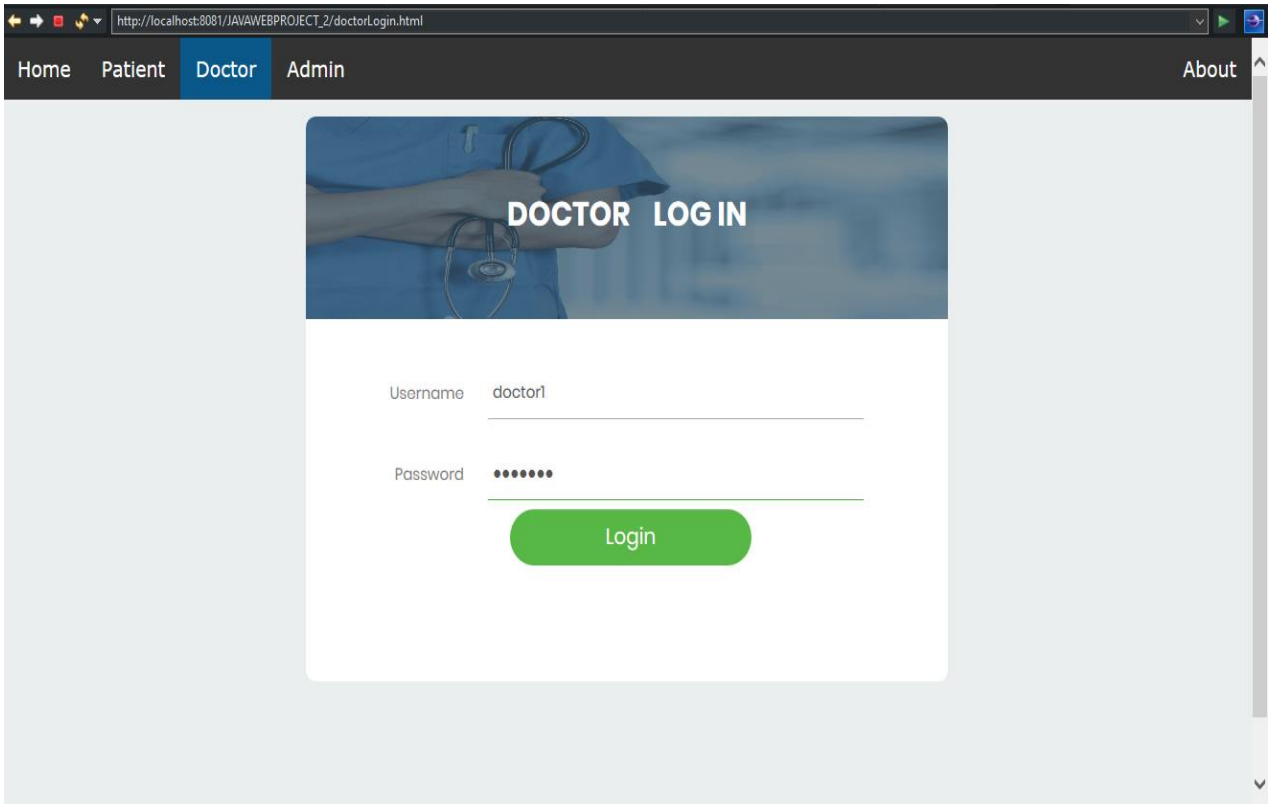
Εμφάνιση κλεισμένου ραντεβού

All Pending Appointments

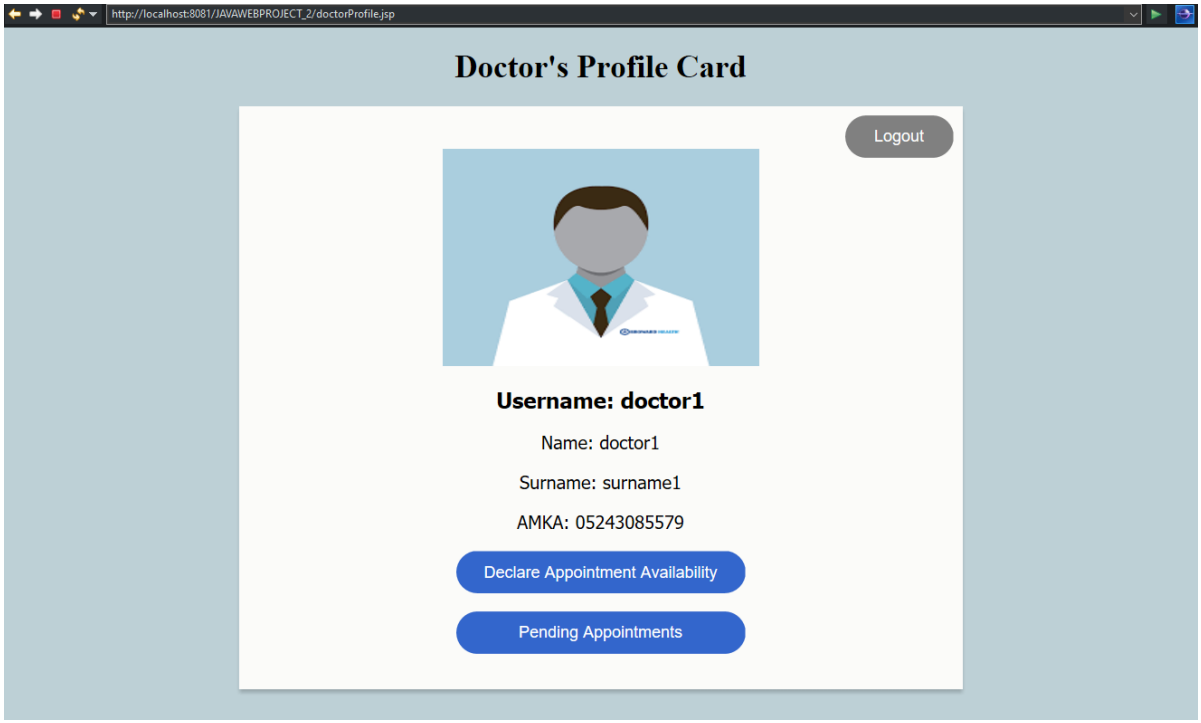
Date	Start Slot Time	End Slot Time	Doctor's AMKA	Doctor's Name	Action
2021-07-14	10:00:00	10:30:00	10192442815	Dimitrios	Delete
2021-07-20	14:00:00	14:30:00	41381050148	Brenn	Delete
2021-07-21	16:00:00	16:30:00	10192442815	Dimitrios	Delete
2021-07-22	17:30:00	18:00:00	10192442815	Dimitrios	Delete
2021-07-26	13:30:00	14:00:00	55670497446	Roger	Delete
2021-07-29	20:00:00	20:30:00	05243085579	doctor1	Delete
2021-08-04	09:00:00	09:30:00	07475555705	Thanos	Delete
2021-08-17	09:00:00	09:30:00	07475555705	Thanos	Delete
2021-08-21	09:30:00	10:00:00	98821388042	Jeffy	Delete
2021-08-30	16:00:00	16:30:00	68968421329	Stace	Delete

4 Λειτουργίες Ιατρών (Doctor):

4.1 Λειτουργία σύνδεσης (login) για τον Ιατρό

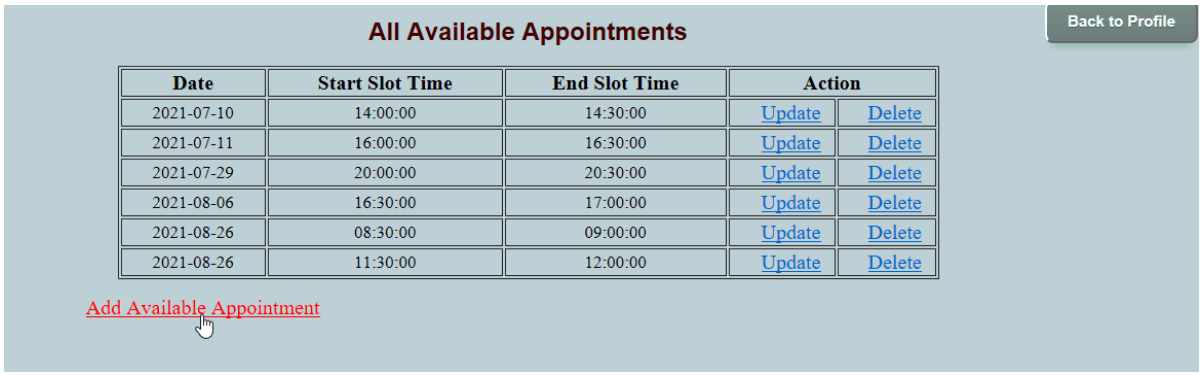


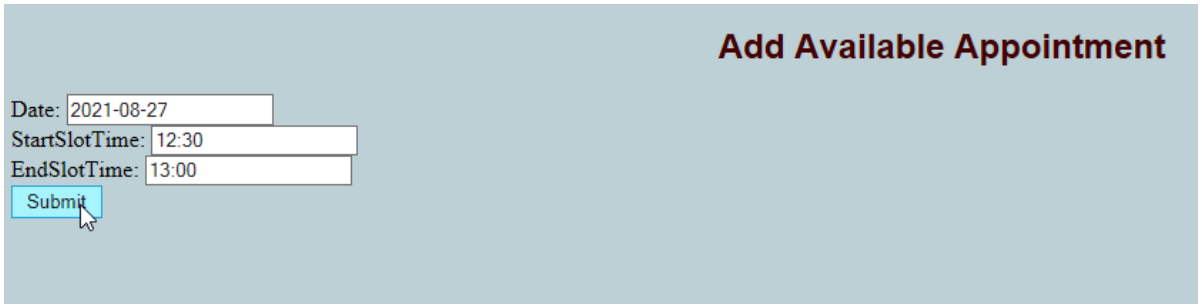
4.2 Προβολή στοιχείων του Ιατρού(μόνο τα στοιχεία του γιατρού που έχει συνδεθεί).

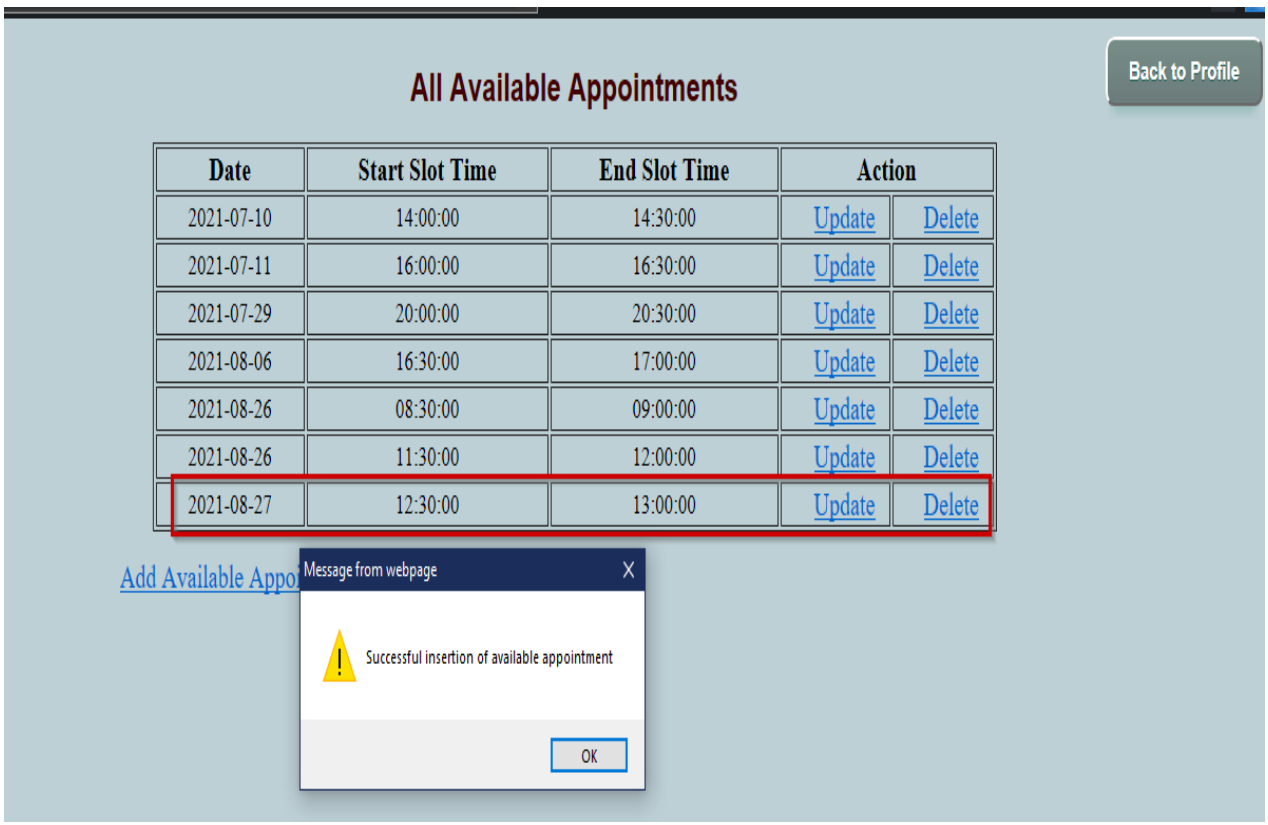


4.3 Καταχώρηση/επεξεργασία/διαγραφή διαθεσιμότητας ραντεβού

4.3.1 Καταχώρηση διαθεσιμότητας ραντεβού





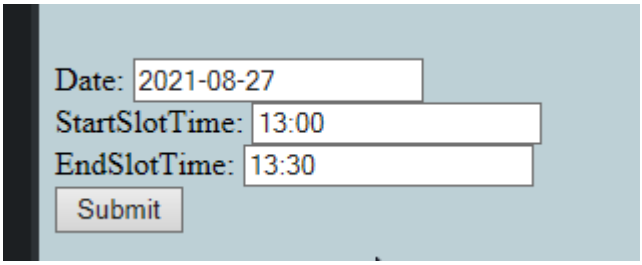


Επιτυχής προσθήκη!

4.3.2 Επεξεργασία(Update) ραντεβού διαθεσιμότητας

Επεξεργασία ώρας για το ραντεβού στις 2021-08-27

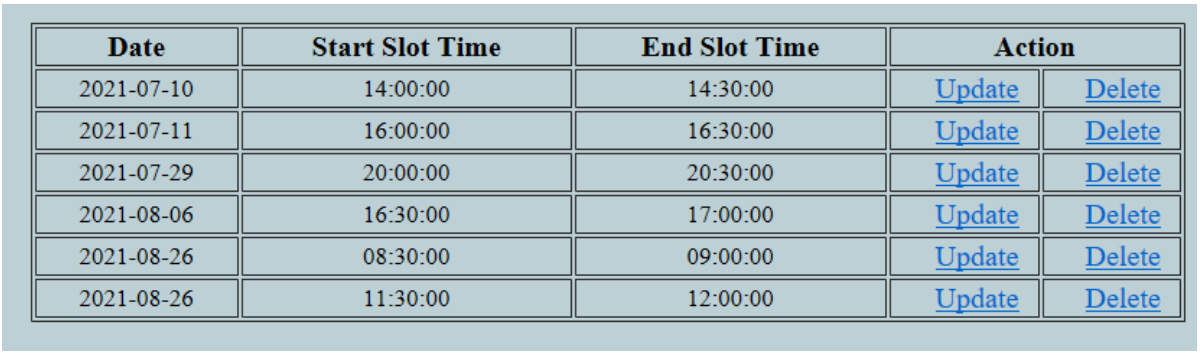
Από 12:30 μεταφορά ώρας στις 13:00



Το EndSlotTime πηγαίνει αυτοματοποημένα μισή ώρα μετά το StartSlotTime

Επιτυχής αλλαγή!

4.3.3 Διαγραφή (Delete) ραντεβού διαθεσιμότητας

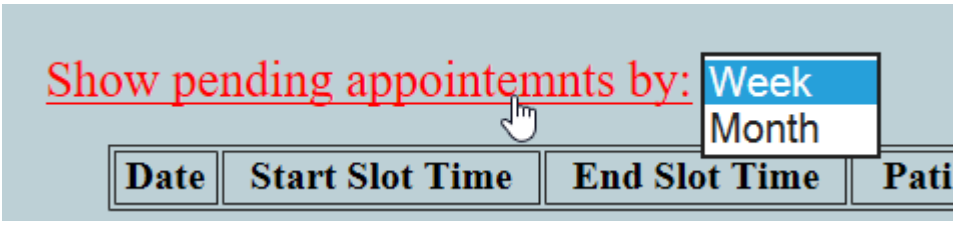


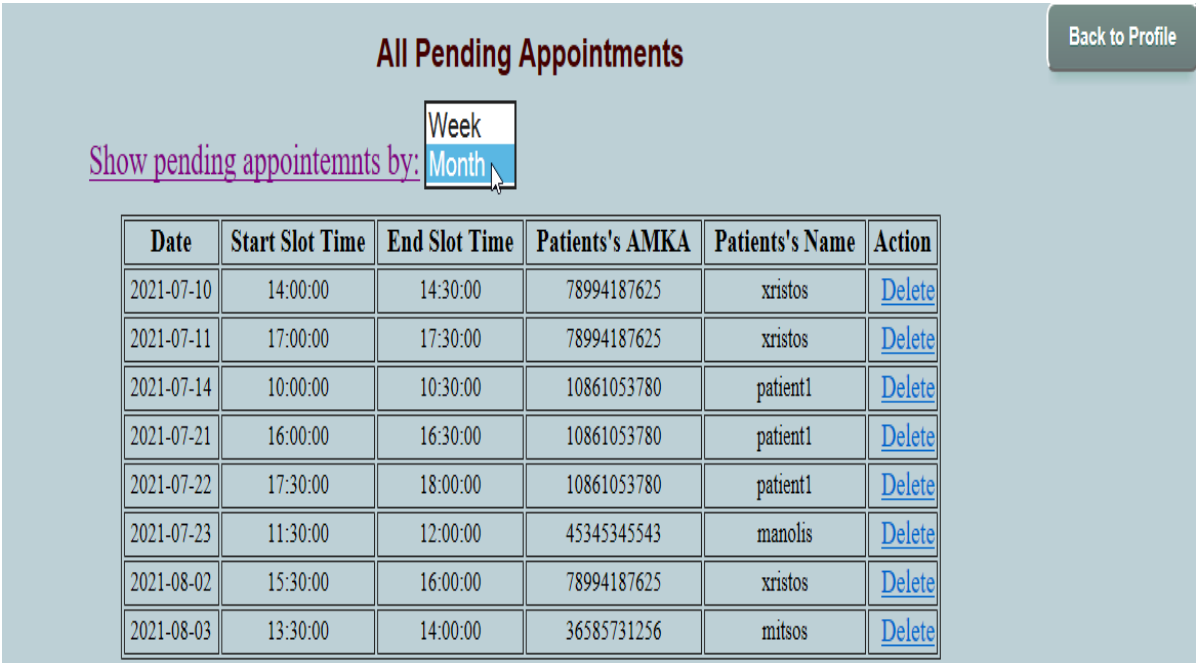
Date	Start Slot Time	End Slot Time	Action	
2021-07-10	14:00:00	14:30:00	Update	Delete
2021-07-11	16:00:00	16:30:00	Update	Delete
2021-07-29	20:00:00	20:30:00	Update	Delete
2021-08-06	16:30:00	17:00:00	Update	Delete
2021-08-26	08:30:00	09:00:00	Update	Delete
2021-08-26	11:30:00	12:00:00	Update	Delete

Επιτυχής διαγραφή!

4.4 Προβολή πίνακα /Διαγραφή ραντεβού

4.4.1 Προβολή

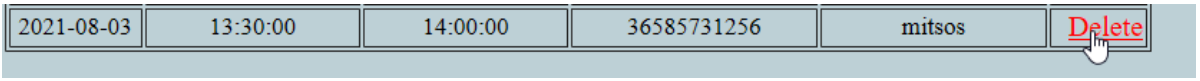




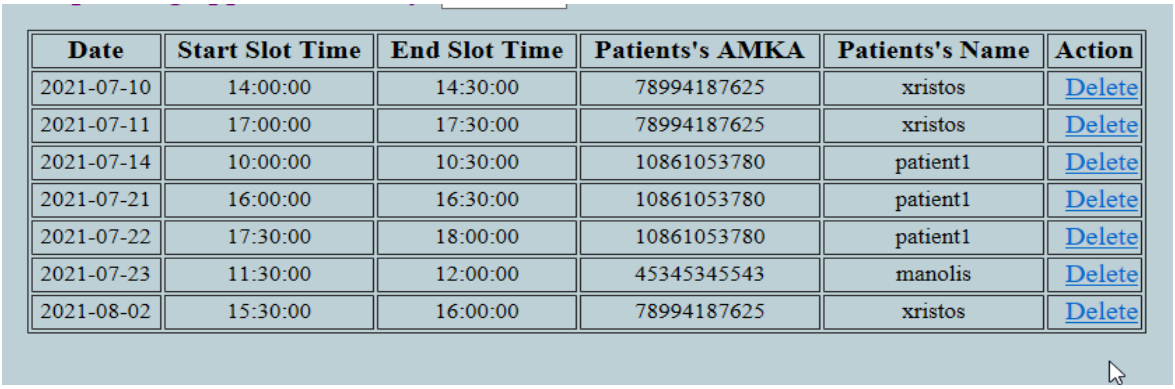
Date	Start Slot Time	End Slot Time	Patients's AMKA	Patients's Name	Action
2021-07-10	14:00:00	14:30:00	78994187625	xristos	Delete
2021-07-11	17:00:00	17:30:00	78994187625	xristos	Delete
2021-07-14	10:00:00	10:30:00	10861053780	patient1	Delete
2021-07-21	16:00:00	16:30:00	10861053780	patient1	Delete
2021-07-22	17:30:00	18:00:00	10861053780	patient1	Delete
2021-07-23	11:30:00	12:00:00	45345345543	manolis	Delete
2021-08-02	15:30:00	16:00:00	78994187625	xristos	Delete
2021-08-03	13:30:00	14:00:00	36585731256	mitsos	Delete

Προβολή Μηνιαίων ή Εβδομαδιαίων ραντεβού

4.4.2 Διαγραφή



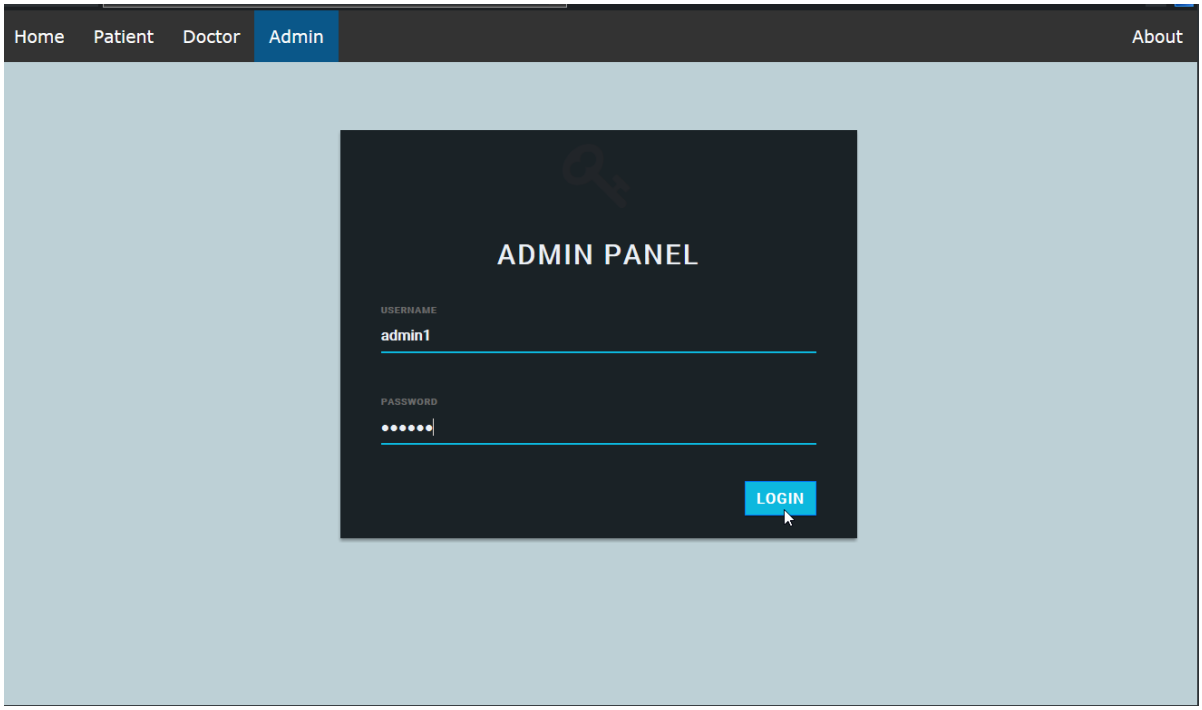
Προβολή πίνακα μετά την διαγραφή



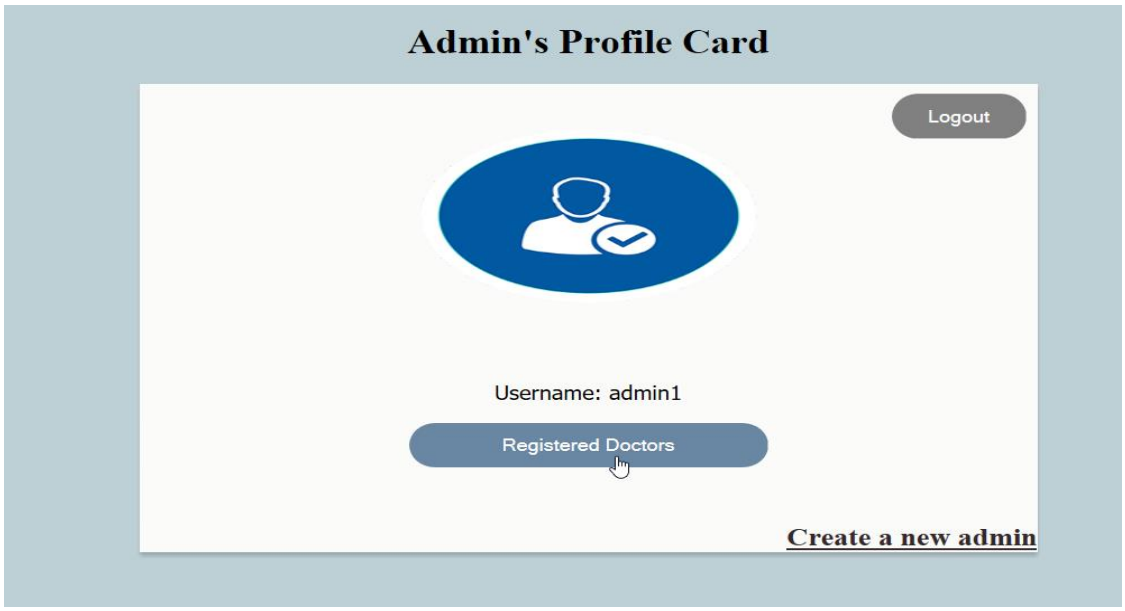
Date	Start Slot Time	End Slot Time	Patients's AMKA	Patients's Name	Action
2021-07-10	14:00:00	14:30:00	78994187625	xristos	Delete
2021-07-11	17:00:00	17:30:00	78994187625	xristos	Delete
2021-07-14	10:00:00	10:30:00	10861053780	patient1	Delete
2021-07-21	16:00:00	16:30:00	10861053780	patient1	Delete
2021-07-22	17:30:00	18:00:00	10861053780	patient1	Delete
2021-07-23	11:30:00	12:00:00	45345345543	manolis	Delete
2021-08-02	15:30:00	16:00:00	78994187625	xristos	Delete

5 Λειτουργίες Διαχειριστή (Administrator):

5.1 Λειτουργία σύνδεσης (login) για τον Διαχειριστή

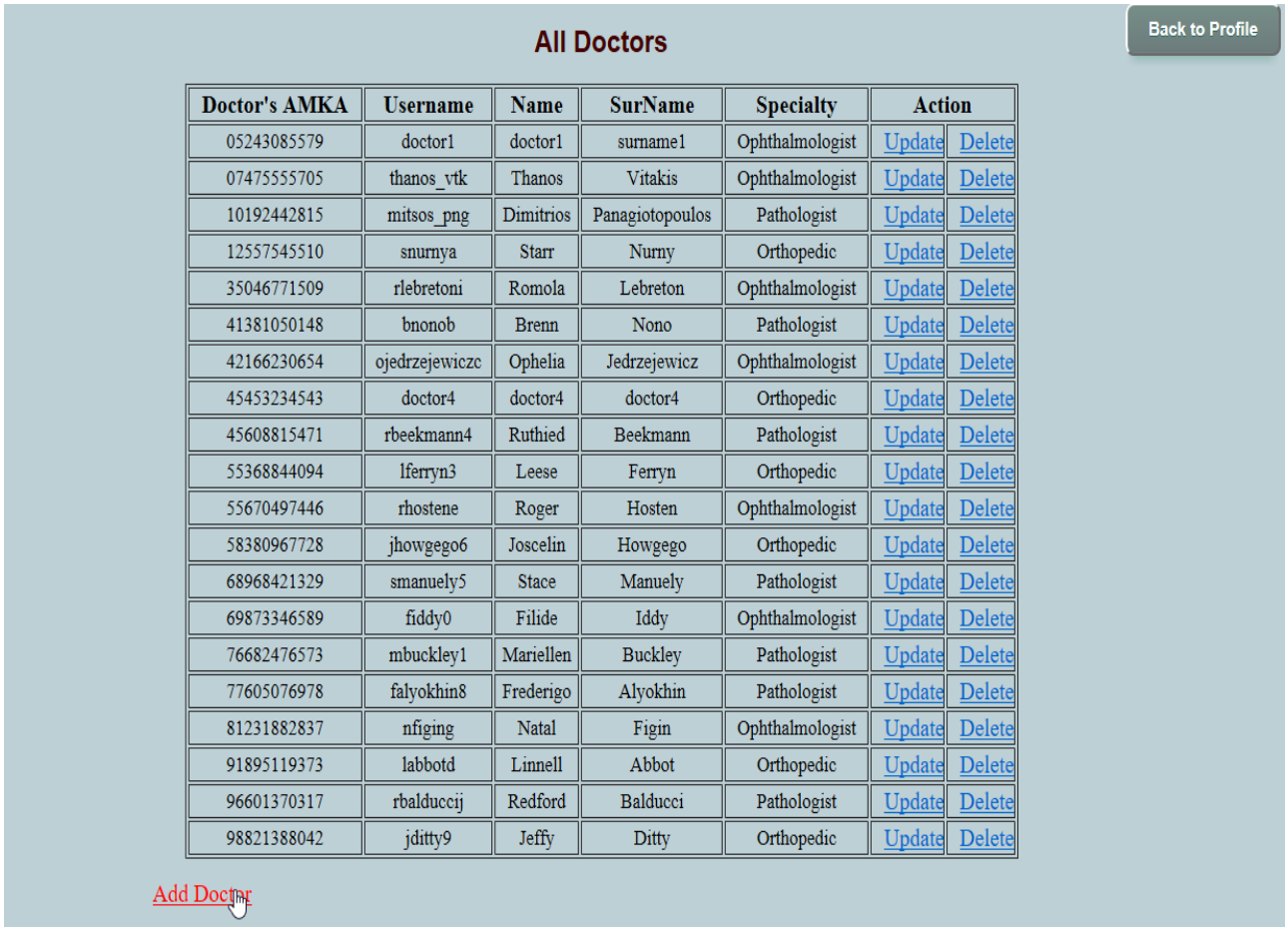


5.2 Προβολή στοιχείων του διαχειριστή(μόνο τα στοιχεία του ασθενούς που έχει συνδεθεί).



5.3 Προβολή/Εισαγωγή/Επεξεργασία/Διαγραφή Ιατρού/ων

5.3.1 Προβολή



Doctor's AMKA	Username	Name	SurName	Specialty	Action	
05243085579	doctor1	doctor1	surname1	Ophthalmologist	Update	Delete
07475555705	thanos_vtk	Thanos	Vitakis	Ophthalmologist	Update	Delete
10192442815	mitsos_png	Dimitrios	Panagiotopoulos	Pathologist	Update	Delete
12557545510	snurnya	Starr	Nurny	Orthopedic	Update	Delete
35046771509	rlebretoni	Romola	Lebreton	Ophthalmologist	Update	Delete
41381050148	bnonob	Brenn	Nono	Pathologist	Update	Delete
42166230654	ojedrzejewiczc	Ophelia	Jedrzejewicz	Ophthalmologist	Update	Delete
45453234543	doctor4	doctor4	doctor4	Orthopedic	Update	Delete
45608815471	rbeekmann4	Ruthied	Beekmann	Pathologist	Update	Delete
55368844094	lferryn3	Leese	Ferryn	Orthopedic	Update	Delete
55670497446	rhostene	Roger	Hosten	Ophthalmologist	Update	Delete
58380967728	jhowgego6	Joscelin	Howgego	Orthopedic	Update	Delete
68968421329	smanuely5	Stace	Manuely	Pathologist	Update	Delete
69873346589	fiddy0	Filide	Iddy	Ophthalmologist	Update	Delete
76682476573	mbuckley1	Mariellen	Buckley	Pathologist	Update	Delete
77605076978	falyokhin8	Frederigo	Alyokhin	Pathologist	Update	Delete
81231882837	nfiging	Natal	Figin	Ophthalmologist	Update	Delete
91895119373	labbotd	Linnell	Abbot	Orthopedic	Update	Delete
96601370317	rbalduccij	Redford	Balducci	Pathologist	Update	Delete
98821388042	jditty9	Jeffy	Ditty	Orthopedic	Update	Delete

Username Mike

Password ••••

First name Michalis

Last name Papadopoulos

AMKA 1234567890

Specialty Pathologist / Ophthalmologist / Orthopedic

Add Doctor

Doctor's AMKA	Username	Name	SurName	Specialty	Action	
05243085579	doctor1	doctor1	surname1	Ophthalmologist	Update	Delete
07475555705	thanos_vtk	Thanos	Vitakis	Ophthalmologist	Update	Delete
10192442815	mitsos_png	Dimitrios	Panagiotopoulos	Pathologist	Update	Delete
1234567890	Mike	Michalis	Papadopoulos	Pathologist	Update	Delete
12557545510	snurnya	Starr	Nurny	Orthopedic	Update	Delete

5.3.3 Επεξεργασία

1234567890	Mike	Michalis	Papadopoulos	Pathologist	Update	Delete

AMKA : 1234567890
Username : Mike

Password:
Name : Mixalis
Surname : Papadopoulos
Choose a Specialty: Pathologist (Pathologist / Ophthalmologist / Orthopedic) Submit

1234567890	Mike	Mixalis	Papadopoulos	Orthopedic	Update	Delete

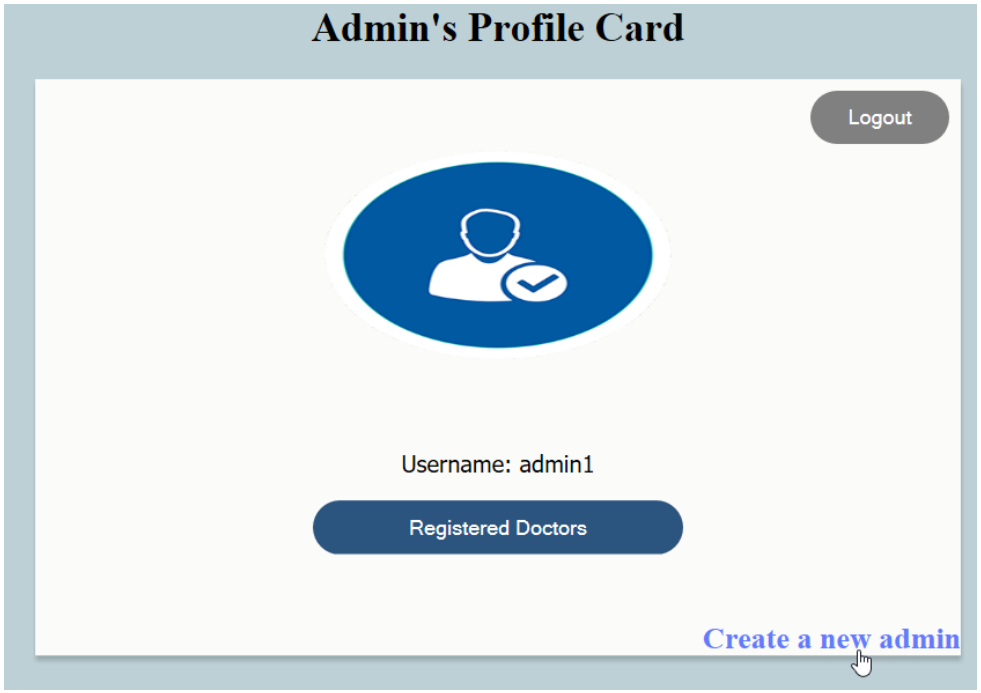
5.3.4 Διαγραφή

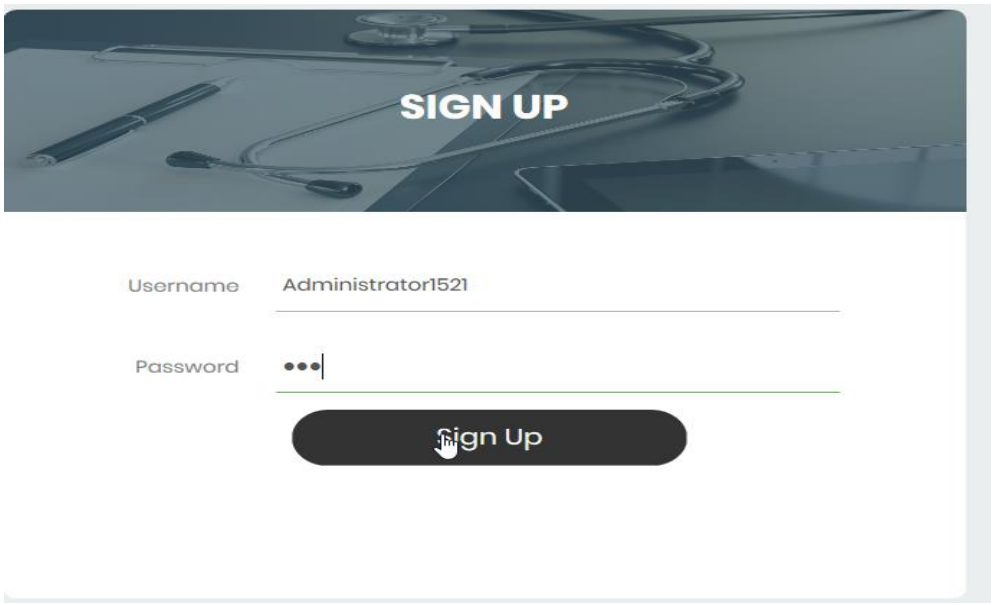
10192442815	mitsos_png	Dimitrios	Panagiotopoulos	Pathologist	Update	Delete
1234567890	Mike	Mixalis	Papadopoulos	Orthopedic	Update	Delete
12557545510	snurnya	Starr	Nurny	Orthopedic	Update	Delete

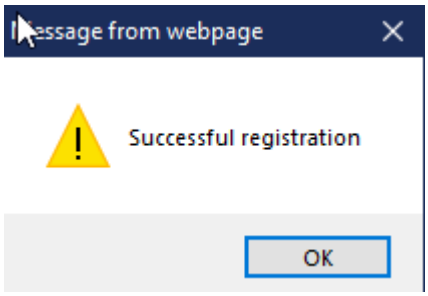
Μετά την διαγραφή

10192442815	mitsos_png	Dimitrios	Panagiotopoulos	Pathologist	Update	Delete
12557545510	snurnya	Starr	Nurny	Orthopedic	Update	Delete

5.4 Extra: Λειτουργία εισαγωγής νέου Διαχειριστή







5.5 Extra:Σελίδα About

6 Κώδικας προγράμματος

6.1 Κώδικας Patient

6.1.1 Κλάση Patient

```
1  package Classes;
2
3  public class Patient extends Users {
4
5
6      /*Constructor*/
7      public Patient(String username, String password, String name, String surname, String amka) {
8          super(username, password, name, surname,amka);
9
0
1      }
2
3
4
5
6      public String registration(){
7          String insertPatientStatement = "INSERT INTO PATIENT (patientAMKA, username, hashedpassword, name, surname, salt) VALUES (?,?,?,?,?,?)";
8          return(insertPatientStatement);
9      }
0
1
2  }
3
```

6.1.2 Patient Servlet

```
1  package Servlets;
2
3  import java.io.IOException;
27
28 @WebServlet("/PatientServlet")
29 public class PatientServlet extends HttpServlet {
30
31    private static final long serialVersionUID = 1L;
32
33
34    public static patientDao dao;
35
36    private static String INSERT_Appointment = "/addPatientAppointment.jsp";
37    private static String LIST_PendingAppointments = "/PatientsListPendingAppointments.jsp";
38
39    public PatientServlet() {
40       super();
41       dao = new patientDao();
42    }
43
44    public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
45       PrintWriter out=response.getWriter();
46       String forward="";
47       String action = request.getParameter("action");
48       response.setContentType("text/html; charset=UTF-8");
49       response.setCharacterEncoding("UTF-8");
50       request.setCharacterEncoding("UTF-8");
51       HttpSession session=request.getSession();
52       boolean tempDelete = false;
53       boolean tempBook = false;
54       if (action.equalsIgnoreCase("delete")){
55          Date sqlDate = null;
56          Time StartSlotTime=null;
57          Time EndSlotTime=null;
58          try {
59             java.util.Date date = new SimpleDateFormat("yyyy-MM-dd").parse( request.getParameter("date"));
60             sqlDate = new java.sql.Date(date.getTime());
61
62             DateFormat formatter = new SimpleDateFormat("HH:mm:ss");
63
```

Όταν ο χρηστης που τρεχει την εφαρμογη πατησει το κουμπι delete/edit/add τοτε από την σελιδα jsp μεταβαινουμε στο αντιστοιχο servlet με μεθοδο GET και ένα συγκεκριμενο action και καποιες παραμετρους.Εκει ταυτοποιουμε το action και παιρνουμε τις παραμετρους από το request.Στη συνεχεια καλουμε την μεθοδο από το dao η οποια θα κανει το αντιστοιχο delete/edit/add.

```
63              StartSlotTime = new Time(formatter.parse(request.getParameter("StartSlotTime")).getTime());
64
65              EndSlotTime = new Time(formatter.parse(request.getParameter("EndSlotTime")).getTime());
66          } catch (ParseException e) {
67              // TODO Auto-generated catch block
68              e.printStackTrace();
69          }
70
71          String patientAMKA = session.getAttribute("patientAMKA").toString();
72          String doctorAMKA =  request.getParameter("doctorAMKA");
73
74          LocalDate tempdate=LocalDate.now();
75          tempdate=tempdate.plusDays(3);
76          Date datenowplus3=Date.valueOf(tempdate);
77
78          //System.out.println(sqlDate+" "+datenowplus3 );
79          if(datenowplus3.before(sqlDate)){
80              Confirmed_appointments conf = new Confirmed_appointments(sqlDate,StartSlotTime,EndSlotTime,patientAMKA,doctorAMKA);
81              dao.deleteAppointment(conf);
82          }else {
83              tempDelete = true;
84          }
85
86          forward = LIST_PendingAppointments;
87          request.setAttribute("PendingAppointments", dao.getAllPendingAppointments(patientAMKA));
88
89      }  else if (action.equalsIgnoreCase("ListPendingAppointments")){
90          forward = LIST_PendingAppointments;
91          String patientAMKA = session.getAttribute("patientAMKA").toString();
92          request.setAttribute("PendingAppointments", dao.getAllPendingAppointments(patientAMKA));
93
94      } else if(action.equalsIgnoreCase("insert")){
95          String specialty =  request.getParameter("specialty");
96          forward = INSERT_Appointment;
97          request.setAttribute("AvailableAppointments", dao.getAvailableAppointmentsBySpecialty(specialty));
98
99      }else if((action.equalsIgnoreCase("book"))) {
100         Date sqlDate = null;
101         Time StartSlotTime=null;
102         Time EndSlotTime = null;
```

```
102         Time EndSlotTime=null;
103         try {
104            java.util.Date date = new SimpleDateFormat("yyyy-MM-dd").parse( request.getParameter("date"));
105            sqlDate = new java.sql.Date(date.getTime());
106
107
108            DateFormat formatter = new SimpleDateFormat("HH:mm:ss");
109            StartSlotTime = new Time(formatter.parse(request.getParameter("StartSlotTime")).getTime());
110
111            EndSlotTime = new Time(formatter.parse(request.getParameter("EndSlotTime")).getTime());
112
113         } catch (ParseException e) {
114            // TODO Auto-generated catch block
115            e.printStackTrace();
116         }
117
118         String doctorAMKA =  request.getParameter("doctorAMKA");
119         String patientAMKA = session.getAttribute("patientAMKA").toString();
120         Confirmed_appointments conf = new Confirmed_appointments(sqlDate,StartSlotTime,EndSlotTime,patientAMKA,doctorAMKA);
121         dao.BookAppointment(conf);
122         tempBook = true;
123         forward = LIST_PendingAppointments;
124         request.setAttribute("PendingAppointments", dao.getAllPendingAppointments(patientAMKA));
125      }
126
127      RequestDispatcher view = request.getRequestDispatcher(forward);
128
129      if(tempDelete) {
130         view.include(request, response);
131         out.println("<script>" + "alert(\"Can't delete appointment because it's less than 3 days later\");" +
132               "</script>");
133      }else if(tempBook){
134         view.include(request, response);
135         out.println("<script>" + "alert(\"Appointment booked succesfully\");" +
136               "</script>");
137      }else {
138         view.forward(request, response);
139      }
140
141   `
```

```
141   }
142   public void doPost(HttpServletRequest req,HttpServletResponse res) throws ServletException,IOException{
143
144      HttpSession session=req.getSession();
145      if(session.getAttribute("doctorAMKA")!=null) {
146         DoctorServlet.Logout(req, res);
147      }else if(session.getAttribute("userid")!=null) {
148         AdminServlet.Logout(req, res);
149      }
150
151      if(req.getParameter("LoginButton")!=null){
152         Login(req,res);
153
154      }else if(req.getParameter("SignupButton")!=null){
155         SignupPatient(req,res);
156      }
157      else if(req.getParameter("LogoutButton")!=null) {
158         Logout(req,res);
159         res.sendRedirect("patientLogin.html");
160      }
161   }
162
163
164   public void Login(HttpServletRequest req,HttpServletResponse res) throws IOException, ServletException{
165      PrintWriter out=res.getWriter();
166      String uname=req.getParameter("username");
167      String pass=req.getParameter("pass");
168      String name=null;
169      String surname=null;
170      String patientAMKA=null;
171
172      String sql="Select * from patient where username=?";
173      boolean found=false;
174      boolean exceptionThrown = false;
175      try {
176
177      PreparedStatement st=dao.connection.prepareStatement(sql);
178      st.setString(1, uname);
179
180
```

Η μεθοδος POST του κάθε servlet ενεργοποιειται όταν ο χρηστης θελει να κανει login,signup,logout η καποιο edit/insert.

```
180
181     ResultSet rs = st.executeQuery();
182
183     boolean next=rs.next();
184     if(next) {
185        String salt=rs.getString("salt");
186        String newhashedpass=Encryption.getHashMD5(pass, salt);
187
188
189        while(next) {
190
191           if (uname.equals(rs.getString("username")) && (newhashedpass.equals(rs.getString("hashedpassword") ))  ){
192              found=true;
193              name=rs.getString("name");
194              surname=rs.getString("surname");
195              patientAMKA=rs.getString("patientAMKA");
196              break;
197           }
198           next=rs.next();
199        }
200     }
201     //rs.close();
202     //con.close();
203     }catch (SQLException sqle) {
204        exceptionThrown = true;
205        req.getRequestDispatcher("patientLogin.html").include(req, res);
206        out.println("<script>" + "alert(\"Database connection problem.\");" +
207              "</script>");
208
209     }
210
211     if(found){
212        HttpSession session=req.getSession();
213        session.setAttribute("username", uname);
214        session.setAttribute("pass", pass);
215        session.setAttribute("name", name);
216        session.setAttribute("surname", surname);
217        session.setAttribute("patientAMKA", patientAMKA);
218
219
```

Στη μεθοδο login αρχικα παιρνουμε το salt που είναι αποθηκευμενο στη βαση με το συγκεκριμενο username που δινει ο χρηστης.Υστερα κανουμε encrypt το password που εχει δωσει ο χρηστης με τη βοηθεια του salt και της κλασης Encryption και ελεγχουμε αν είναι ιδιο με αυτό που βρισκεται στη βαση.Αν είναι ιδιο σημαινει ότι ο χρηστης εδωσε σωστα στοιχεια και τοτε δημιουργουμε το session θετοντας ως παραμετρους τα χαρακτηριστικα του συγκεκριμενου χρηστη που εκανε login(ειτε είναι patient,doctor η admin).

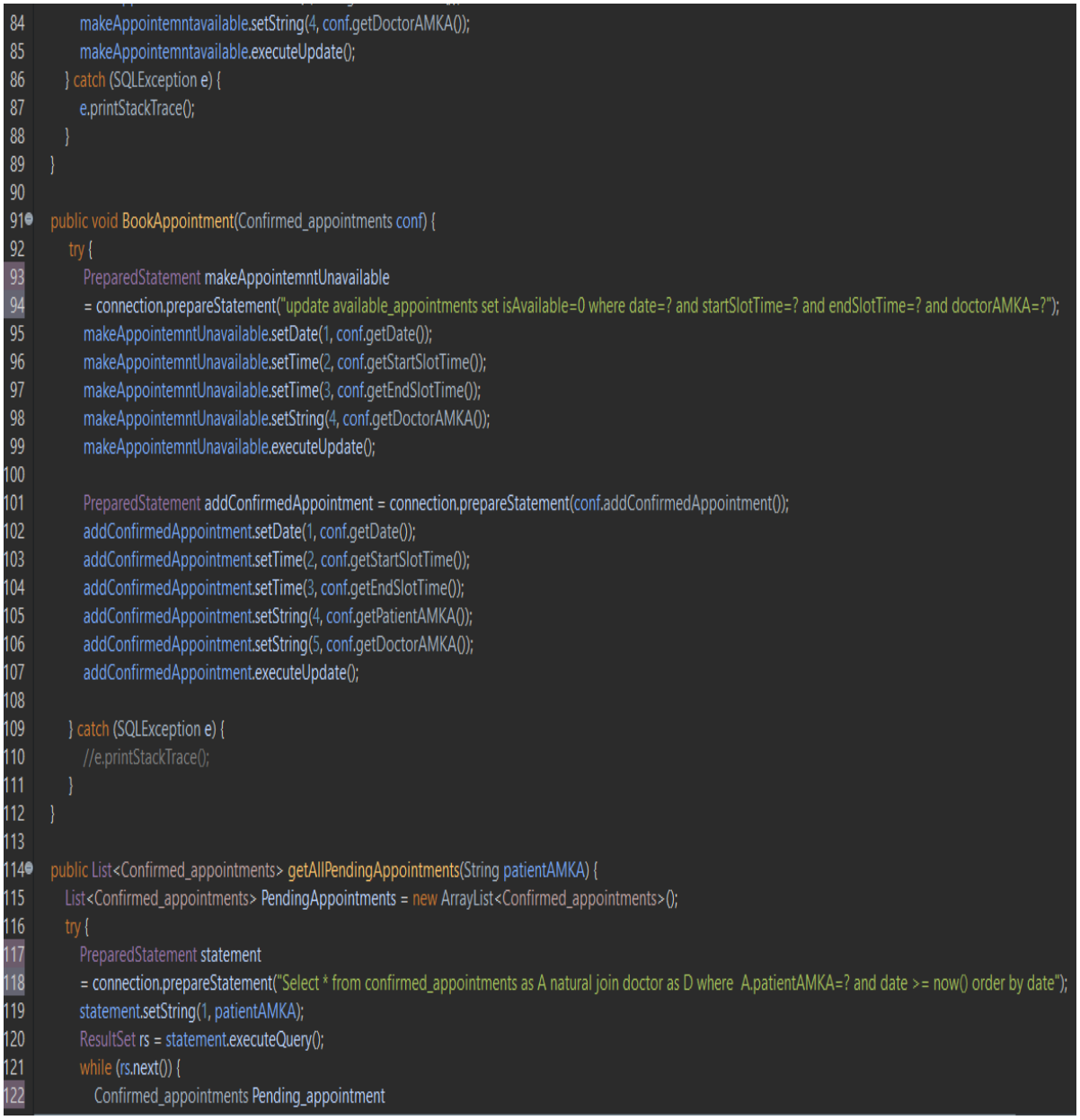
```
219
220         res.sendRedirect("patientProfile.jsp");
221       }else {
222         if(!exceptionThrown) {
223
224             req.getRequestDispatcher("patientLogin.html").include(req, res);
225             out.println("<script>" + "alert(\"Account wasn't found\");" +
226                     "</script>");
227         }
228       }
229       out.flush();
230       out.close();
231   }
232
233
234
235
236   public void SignupPatient(HttpServletRequest req,HttpServletResponse res) throws IOException, ServletException{
237         PrintWriter out=res.getWriter();
238         String patientAMKA = req.getParameter("AMKA");
239         String username = req.getParameter("username");
240         String password = req.getParameter("pass");
241         String name = req.getParameter("First name");
242         String surname = req.getParameter("Last name");
243         if(!isNullOrBlankOrWhiteSpace(patientAMKA)&&!isNullOrBlankOrWhiteSpace(username)
244                 &&!isNullOrBlankOrWhiteSpace(password)&&
245                 !isNullOrBlankOrWhiteSpace(name)&&!isNullOrBlankOrWhiteSpace(surname)){
246             Patient patient = new Patient(username,password,name,surname,patientAMKA);
247             dao.addPatient(patient, req, res);
248
249         }else {
250
251             req.getRequestDispatcher("SignUpPatient.html").include(req, res);
252             out.println("<script>" + "alert(\"Inputs  cannot be blank or contain white spaces\");" +
253                     "</script>");
254
255         }
256
257          out.flush();
258
```

```
258             out.close();
259         }
260
261         public static boolean isNullOrBlankOrWhiteSpace(String value) {
262
263             if (value == null) {
264                 return true;
265             }
266             if (value.length() == 0) {
267                 return true;
268             }
269             for (int i = 0; i < value.length(); i++) {
270                 if (Character.isWhitespace(value.charAt(i))) {
271                     return true;
272                 }
273             }
274
275             return false;
276         }
277
278
279         public static void Logout(HttpServletRequest req,HttpServletResponse res) throws ServletException,IOException{
280
281         HttpSession session=req.getSession();
282         session.removeAttribute("username");
283         session.removeAttribute("pass");
284         session.removeAttribute("name");
285         session.removeAttribute("surname");
286         session.removeAttribute("patientAMKA");
287         session.invalidate();
288         }
289     }
290
```

6.1.3 Patient Dao

```
package dao;

import java.io.IOException;

public class patientDao {
    public Connection connection;

    public patientDao() {
        connection = DbUtil.getConnection();
    }

    public void addPatient(Patient patient,HttpServletRequest req,HttpServletResponse res) throws IOException, ServletException {
        PrintWriter out=res.getWriter();
        try {

            String salt=Encryption.createSalt().toString();
            PreparedStatement insertPatient = connection.prepareStatement(patient.registration());
            insertPatient.setString(1,patient.getAMKA());
            insertPatient.setString(2, patient.getUsername());
            insertPatient.setString(3, Encryption.getHashMD5(patient.getPassword(),salt));
            insertPatient.setString(4, patient.getName());
            insertPatient.setString(5, patient.getSurname());
            insertPatient.setString(6, salt.toString());
            insertPatient.executeUpdate();

            //insertPatient.close();
            //con.close();
```

```


            out.println("<script>" + "alert(\"Successful registration\");" +
            "</script>");
            req.getRequestDispatcher("patientLogin.html").include(req, res);

        } catch(SQLException sqle ) {
            //sqle.printStackTrace();
            if(sqle.getMessage().contains(patient.getAMKA())) {

                req.getRequestDispatcher("SignUpPatient.html").include(req, res);
                out.println("<script>" + "alert(\"Wrong AMKA.Please give your real AMKA.\");" +
                        "</script>");

            }
            else if(sqle.getMessage().contains(patient.getUsername())){

                req.getRequestDispatcher("SignUpPatient.html").include(req, res);
                out.println("<script>" + "alert(\"This username is already in use.Please try an other username.\");" +
                        "</script>");

            }

        }
    }

    public void deleteAppointment(Confirmed_appointments conf) {
        try {
            PreparedStatement deleteAppointment
            = connection.prepareStatement("delete from confirmed_appointments where date=? and patientAMKA=? and doctorAMKA=?");
            deleteAppointment.setDate(1, conf.getDate());
            deleteAppointment.setString(2, conf.getPatientAMKA());
            deleteAppointment.setString(3, conf.getDoctorAMKA());
            deleteAppointment.executeUpdate();

            PreparedStatement makeAppointemntavailable
            = connection.prepareStatement("update available_appointments set isAvailable=1 where date=? and startSlotTime=? and endSlotTime=? and doctorAMKA=?");
            makeAppointemntavailable.setDate(1, conf.getDate());
            makeAppointemntavailable.setTime(2, conf.getStartSlotTime());
```



```
84          makeAppointemntavailable.setString(4, conf.getDoctorAMKA());
85          makeAppointemntavailable.executeUpdate();
86        } catch (SQLException e) {
87          e.printStackTrace();
88        }
89      }
90
91      public void BookAppointment(Confirmed_appointments conf) {
92        try {
93          PreparedStatement makeAppointemntUnavailable
94          = connection.prepareStatement("update available_appointments set isAvailable=0 where date=? and startSlotTime=? and endSlotTime=? and doctorAMKA=?");
95          makeAppointemntUnavailable.setDate(1, conf.getDate());
96          makeAppointemntUnavailable.setTime(2, conf.getStartSlotTime());
97          makeAppointemntUnavailable.setTime(3, conf.getEndSlotTime());
98          makeAppointemntUnavailable.setString(4, conf.getDoctorAMKA());
99          makeAppointemntUnavailable.executeUpdate();
100
101         PreparedStatement addConfirmedAppointment = connection.prepareStatement(conf.addConfirmedAppointment());
102         addConfirmedAppointment.setDate(1, conf.getDate());
103         addConfirmedAppointment.setTime(2, conf.getStartSlotTime());
104         addConfirmedAppointment.setTime(3, conf.getEndSlotTime());
105         addConfirmedAppointment.setString(4, conf.getPatientAMKA());
106         addConfirmedAppointment.setString(5, conf.getDoctorAMKA());
107         addConfirmedAppointment.executeUpdate();
108
109       } catch (SQLException e) {
110         //e.printStackTrace();
111       }
112     }
113
114     public List<Confirmed_appointments> getAllPendingAppointments(String patientAMKA) {
115       List<Confirmed_appointments> PendingAppointments = new ArrayList<Confirmed_appointments>();
116       try {
117         PreparedStatement statement
118         = connection.prepareStatement("Select * from confirmed_appointments as A natural join doctor as D where  A.patientAMKA=? and date >= now() order by date");
119         statement.setString(1, patientAMKA);
120         ResultSet rs = statement.executeQuery();
121         while (rs.next()) {
122           Confirmed_appointments Pending_appointment
```

```
            Confirmed_appointments Pending_appointment
            = new Confirmed_appointments(rs.getDate("date"),rs.getTime("startSlotTime"),rs.getTime("endSlotTime"),rs.getString("patientAMKA"),rs.getString("doctorAMKA"));
            Pending_appointment.setDoctorNAME(rs.getString("name"));
            PendingAppointments.add(Pending_appointment);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }

    return PendingAppointments;
}

public List<Available_appointments> getAvailableAppointmentsBySpecialty(String specialty) {
    List<Available_appointments> AvailableAppointments = new ArrayList<Available_appointments>();
    try {
        PreparedStatement Appointments
        = connection.prepareStatement("Select * from available_appointments as A natural join doctor as D where date >= now() and isAvailable=1 and specialty=? order by date");
        Appointments.setString(1, specialty);
        ResultSet rs = Appointments.executeQuery();
        while (rs.next()) {
            Available_appointments Available_appointment
            = new Available_appointments(rs.getDate("date"),rs.getTime("startSlotTime"),rs.getTime("endSlotTime"),rs.getString("doctorAMKA"));
            Available_appointment.setDoctorNAME(rs.getString("name"));
            Available_appointment.setisAvailable(rs.getBoolean("isAvailable"));
            AvailableAppointments.add(Available_appointment);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }

    return AvailableAppointments;
}
}
```

6.2 Κώδικας Doctor

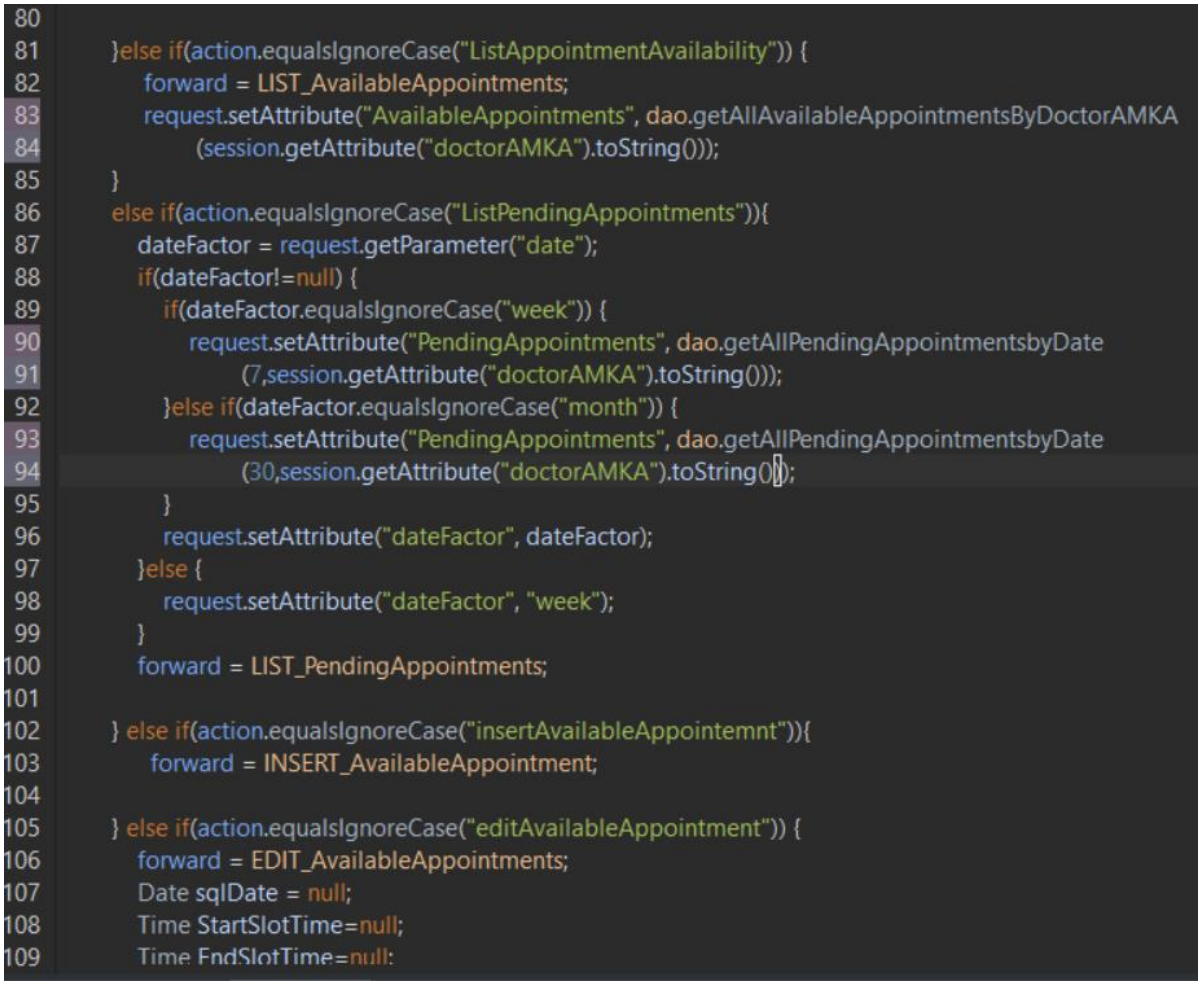
6.2.1 Κλάση Doctor

```
1  package Classes;
2
3  import java.util.ArrayList;
4
5  public class Doctor extends Users {
6
7
8     private final String specialty;
9     private int rate=0;//Ratings will be increasing/decreasing by patients rates
10
11    /*Constructor*/
12    public Doctor(String username, String password, String name, String surname,String specialty,String amka) {
13       super(username, password, name, surname,amka);
14       this.specialty=specialty;
15
16    }
17    public String registration(){
18       String insertDoctorStatement =
19             "INSERT INTO DOCTOR (doctorAMKA, username, hashedpassword, name, surname, specialty, ADMIN_userid, salt) VALUES (?,?,?,?,?,?,?,?)";
20       return(insertDoctorStatement);
```

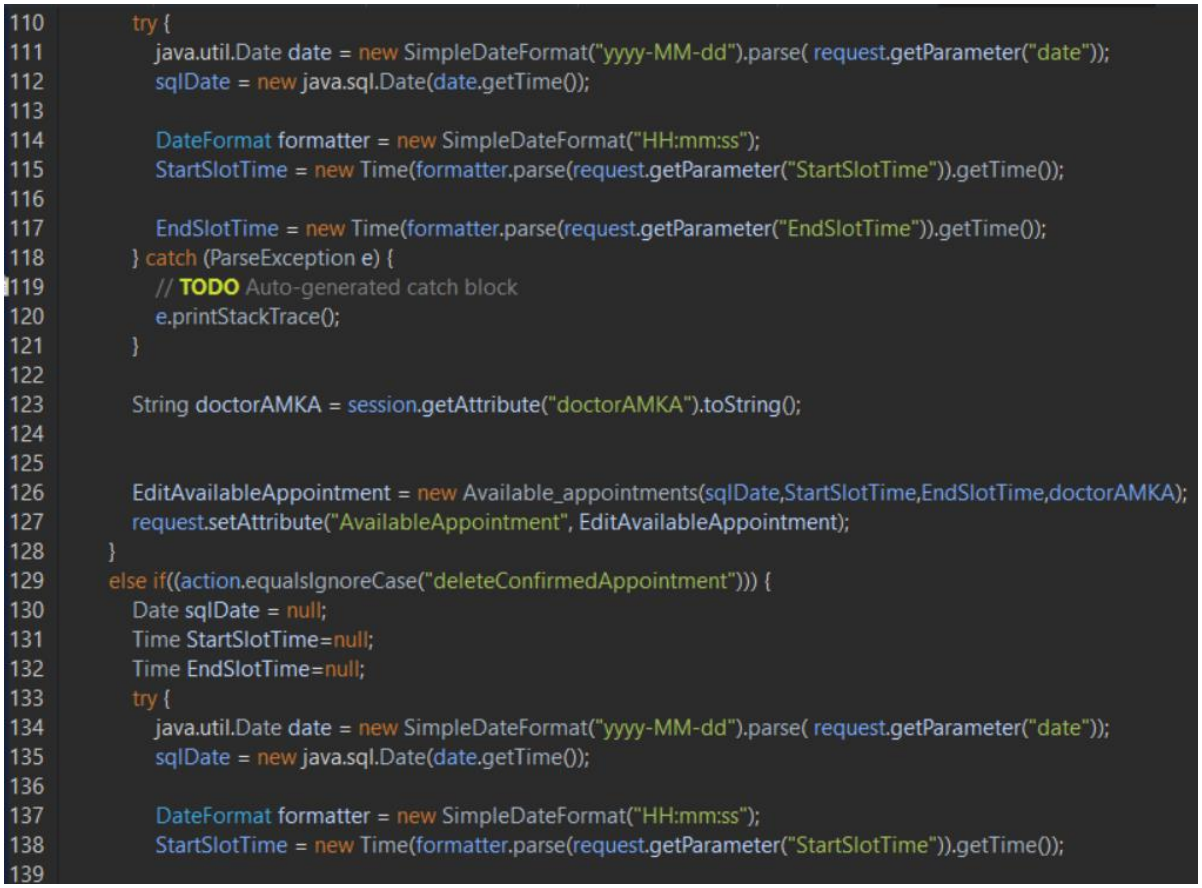
6.2.2 Doctor Servlet

```
1  package Servlets;
2
3  import java.io.IOException;
26
27 @WebServlet("/DoctorServlet")
28 public class DoctorServlet extends HttpServlet {
29
30    private static final long serialVersionUID = 1L;
31
32    public static doctorDao dao;
33    private static String INSERT_AvailableAppointment = "/insertAvailableAppointment.jsp";
34    private static String LIST_PendingAppointments = "/DoctorsListPendingAppointments.jsp";
35    private static String LIST_AvailableAppointments = "/ListAppointmentAvailability.jsp";
36    private static String EDIT_AvailableAppointments = "/EditAppointmentAvailability.jsp";
37
38    String dateFactor = null;
39    Available_appointments EditAvailableAppointment=null;
40
41    public DoctorServlet() {
42       super();
43       dao = new doctorDao();
44    }
45
46
47    public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
48       PrintWriter out=response.getWriter();
49       String forward="";
50       String action = request.getParameter("action");
51       response.setContentType("text/html; charset=UTF-8");
```

```
52       response.setCharacterEncoding("UTF-8");
53       request.setCharacterEncoding("UTF-8");
54       HttpSession session=request.getSession();
55       boolean tempDelete = false;
56
57       if (action.equalsIgnoreCase("deleteAvailableAppointment")){
58
59          Date sqlDate = null;
60          Time StartSlotTime=null;
61          Time EndSlotTime=null;
62          try {
63             java.util.Date date = new SimpleDateFormat("yyyy-MM-dd").parse( request.getParameter("date"));
64             sqlDate = new java.sql.Date(date.getTime());
65
66             DateFormat formatter = new SimpleDateFormat("HH:mm:ss");
67             StartSlotTime = new Time(formatter.parse(request.getParameter("StartSlotTime")).getTime());
68
69             EndSlotTime = new Time(formatter.parse(request.getParameter("EndSlotTime")).getTime());
70          } catch (ParseException e) {
71             e.printStackTrace();
72          }
73
74          String doctorAMKA = session.getAttribute("doctorAMKA").toString();
75          Available_appointments AvailableAppointment = new Available_appointments(sqlDate,StartSlotTime,EndSlotTime,doctorAMKA);
76          dao.deleteAvailableAppointment(AvailableAppointment);
77          forward = LIST_AvailableAppointments;
78          request.setAttribute("AvailableAppointments", dao.getAllAvailableAppointmentsByDoctorAMKA
79                (session.getAttribute("doctorAMKA").toString()));
```

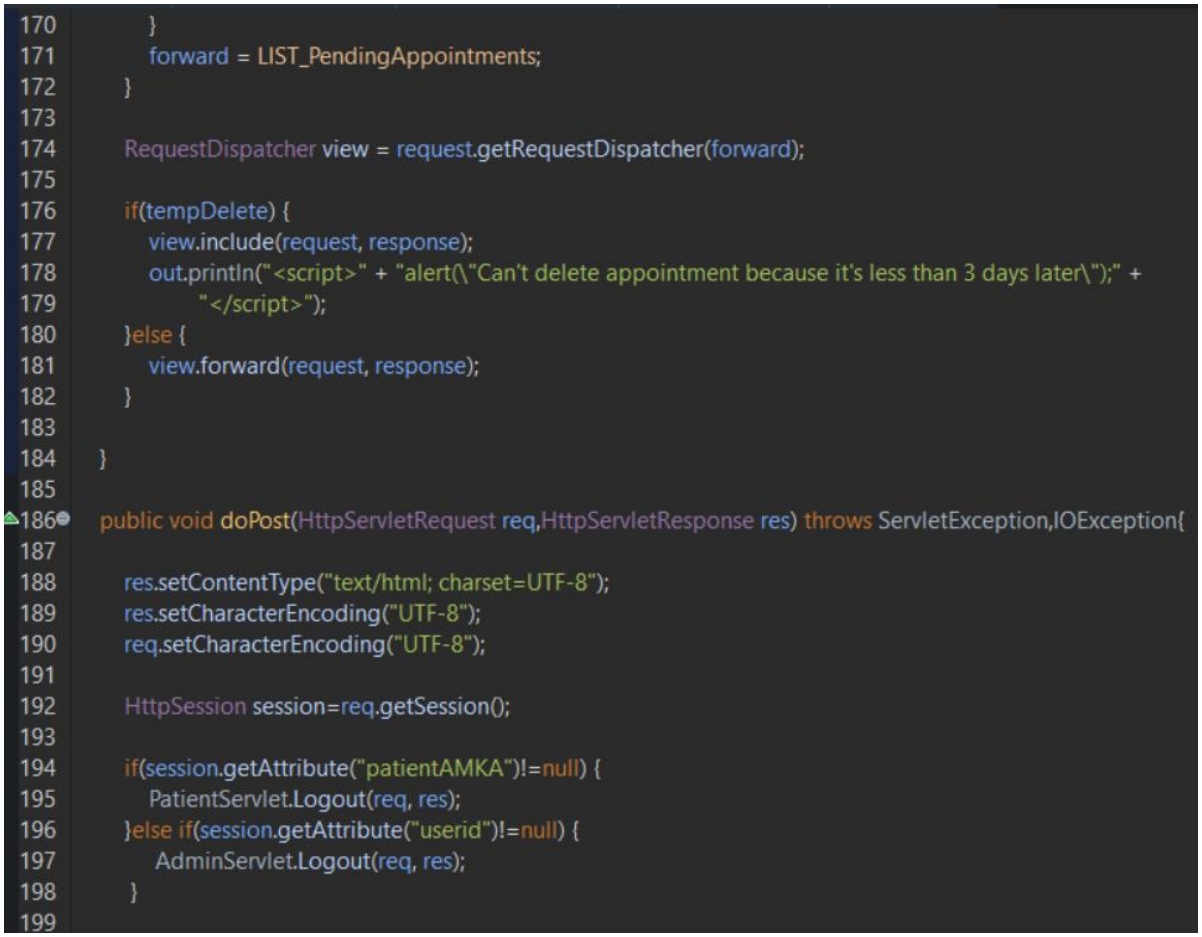


```
80
81      }else if(action.equalsIgnoreCase("ListAppointmentAvailability")) {
82          forward = LIST_AvailableAppointments;
83          request.setAttribute("AvailableAppointments", dao.getAllAvailableAppointmentsByDoctorAMKA
84                  (session.getAttribute("doctorAMKA").toString()));
85      }
86      else if(action.equalsIgnoreCase("ListPendingAppointments")){
87          dateFactor = request.getParameter("date");
88          if(dateFactor!=null) {
89              if(dateFactor.equalsIgnoreCase("week")) {
90                  request.setAttribute("PendingAppointments", dao.getAllPendingAppointmentsbyDate
91                          (7,session.getAttribute("doctorAMKA").toString()));
92              }else if(dateFactor.equalsIgnoreCase("month")) {
93                  request.setAttribute("PendingAppointments", dao.getAllPendingAppointmentsbyDate
94                          (30,session.getAttribute("doctorAMKA").toString());
95              }
96              request.setAttribute("dateFactor", dateFactor);
97          }else {
98              request.setAttribute("dateFactor", "week");
99          }
100         forward = LIST_PendingAppointments;
101
102     } else if(action.equalsIgnoreCase("insertAvailableAppointemnt")){
103         forward = INSERT_AvailableAppointment;
104
105     } else if(action.equalsIgnoreCase("editAvailableAppointment")) {
106         forward = EDIT_AvailableAppointments;
107         Date sqlDate = null;
108         Time StartSlotTime=null;
109         Time EndSlotTime=null;
```



```
110         try {
111            java.util.Date date = new SimpleDateFormat("yyyy-MM-dd").parse( request.getParameter("date"));
112            sqlDate = new java.sql.Date(date.getTime());
113
114            DateFormat formatter = new SimpleDateFormat("HH:mm:ss");
115            StartSlotTime = new Time(formatter.parse(request.getParameter("StartSlotTime")).getTime());
116
117            EndSlotTime = new Time(formatter.parse(request.getParameter("EndSlotTime")).getTime());
118         } catch (ParseException e) {
119            // TODO Auto-generated catch block
120            e.printStackTrace();
121         }
122
123         String doctorAMKA = session.getAttribute("doctorAMKA").toString();
124
125
126         EditAvailableAppointment = new Available_appointments(sqlDate,StartSlotTime,EndSlotTime,doctorAMKA);
127         request.setAttribute("AvailableAppointment", EditAvailableAppointment);
128      }
129      else if((action.equalsIgnoreCase("deleteConfirmedAppointment"))) {
130         Date sqlDate = null;
131         Time StartSlotTime=null;
132         Time EndSlotTime=null;
133         try {
134            java.util.Date date = new SimpleDateFormat("yyyy-MM-dd").parse( request.getParameter("date"));
135            sqlDate = new java.sql.Date(date.getTime());
136
137            DateFormat formatter = new SimpleDateFormat("HH:mm:ss");
138            StartSlotTime = new Time(formatter.parse(request.getParameter("StartSlotTime")).getTime());
139
```

```
140            EndSlotTime = new Time(formatter.parse(request.getParameter("EndSlotTime")).getTime());
141         } catch (ParseException e) {
142            // TODO Auto-generated catch block
143            e.printStackTrace();
144         }
145
146         String doctorAMKA = session.getAttribute("doctorAMKA").toString();
147         String patientAMKA = request.getParameter("patientAMKA");
148
149         LocalDate tempdate=LocalDate.now();
150         tempdate=tempdate.plusDays(3);
151         Date datenowplus3=Date.valueOf(tempdate);//cast
152
153         //System.out.println(sqlDate+" "+datenowplus3 );
154         if(datenowplus3.before(sqlDate)){
155            Confirmed_appointments conf = new Confirmed_appointments(sqlDate,StartSlotTime,EndSlotTime,patientAMKA,doctorAMKA);
156            dao.deleteConfirmedAppointment(conf);
157         }else {
158            tempDelete = true;
159         }
160
161         if(dateFactor!=null) {
162            if(dateFactor.equalsIgnoreCase("week")) {
163               request.setAttribute("PendingAppointments", dao.getAllPendingAppointmentsbyDate(7,session.getAttribute("doctorAMKA").toString()));
164            }else if(dateFactor.equalsIgnoreCase("month")) {
165               request.setAttribute("PendingAppointments", dao.getAllPendingAppointmentsbyDate(30,session.getAttribute("doctorAMKA").toString()));
166            }
167            request.setAttribute("dateFactor", dateFactor);
168         }else {
169            request.setAttribute("dateFactor", "week");
```



```
170         }
171         forward = LIST_PendingAppointments;
172      }
173
174      RequestDispatcher view = request.getRequestDispatcher(forward);
175
176      if(tempDelete) {
177         view.include(request, response);
178         out.println("<script>" + "alert(\"Can't delete appointment because it's less than 3 days later\");" +
179               "</script>");
180      }else {
181         view.forward(request, response);
182      }
183
184   }
185
186   public void doPost(HttpServletRequest req,HttpServletResponse res) throws ServletException,IOException{
187
188      res.setContentType("text/html; charset=UTF-8");
189      res.setCharacterEncoding("UTF-8");
190      req.setCharacterEncoding("UTF-8");
191
192      HttpSession session=req.getSession();
193
194      if(session.getAttribute("patientAMKA")!=null) {
195         PatientServlet.Logout(req, res);
196      }else if(session.getAttribute("userid")!=null) {
197          AdminServlet.Logout(req, res);
198       }
199
```

```
200
201     if(req.getParameter("LoginButton")!=null){
202       Login(req,res);
203
204     }
205     else if(req.getParameter("LogoutButton")!=null) {
206       Logout(req,res);
207       res.sendRedirect("doctorLogin.html");
208     }else if (req.getParameter("editAvailableAppointment")!=null){
209
210       Date sqlDate = null;
211       Time StartSlotTime=null;
212       Time EndSlotTime=null;
213
214       try {
215         java.util.Date date = new SimpleDateFormat("yyyy-MM-dd").parse( req.getParameter("date"));
216         sqlDate = new java.sql.Date(date.getTime());
217
218         DateFormat formatter = new SimpleDateFormat("HH:mm");
219         StartSlotTime = new Time(formatter.parse(req.getParameter("StartSlotTime")).getTime());
220
221         EndSlotTime = new Time(formatter.parse(req.getParameter("EndSlotTime")).getTime());
222
223       } catch (ParseException e) {
224         e.printStackTrace();
225       }
226       String doctorAMKA = session.getAttribute("doctorAMKA").toString();
227       Available_appointments av_appointment = new Available_appointments(sqlDate,StartSlotTime,EndSlotTime,doctorAMKA);
228       if(EditAvailableAppointment!=null) {
229         dao.updateAvailableAppointment(av_appointment,EditAvailableAppointment);
```

```
230       }
231       RequestDispatcher view = req.getRequestDispatcher(LIST_AvailableAppointments);
232       req.setAttribute("AvailableAppointments", dao.getAllAvailableAppointmentsByDoctorAMKA(session.getAttribute("doctorAMKA").toString()));
233       view.forward(req, res);
234     }else if (req.getParameter("insertAvailableAppointment")!=null){
235       Date sqlDate = null;
236       Time StartSlotTime=null;
237       Time EndSlotTime=null;
238
239       try {
240         java.util.Date date = new SimpleDateFormat("yyyy-MM-dd").parse( req.getParameter("date"));
241         sqlDate = new java.sql.Date(date.getTime());
242
243         DateFormat formatter = new SimpleDateFormat("HH:mm");
244         StartSlotTime = new Time(formatter.parse(req.getParameter("StartSlotTime")).getTime());
245
246         EndSlotTime = new Time(formatter.parse(req.getParameter("EndSlotTime")).getTime());
247
248       } catch (ParseException e) {
249         e.printStackTrace();
250       }
251       String doctorAMKA = session.getAttribute("doctorAMKA").toString();
252       Available_appointments av_appointment = new Available_appointments(sqlDate,StartSlotTime,EndSlotTime,doctorAMKA);
253       dao.insertAvailableAppointment(av_appointment,req,res);
254
255     }
256
257   }
258
```

```
259  public void Login(HttpServletRequest req,HttpServletResponse res) throws IOException, ServletException{
260      PrintWriter out=res.getWriter();
261      String uname=req.getParameter("username");
262      String pass=req.getParameter("pass");
263      String name=null;
264      String surname=null;
265      String doctorAMKA=null;
266
267      String sql="Select * from doctor where username=?";
268      boolean found=false;
269      boolean exceptionThrown = false;
270      try {
271
272      PreparedStatement st=dao.connection.prepareStatement(sql);
273      st.setString(1, uname);
274
275
276      ResultSet rs = st.executeQuery();
277
278      boolean next=rs.next();
279
280       if(next) {
281          String salt=rs.getString("salt");
282          String newhashedpass=Encryption.getHashMD5(pass, salt);
283          //System.out.println(salt + " = " + newhashedpass);
284
285
```

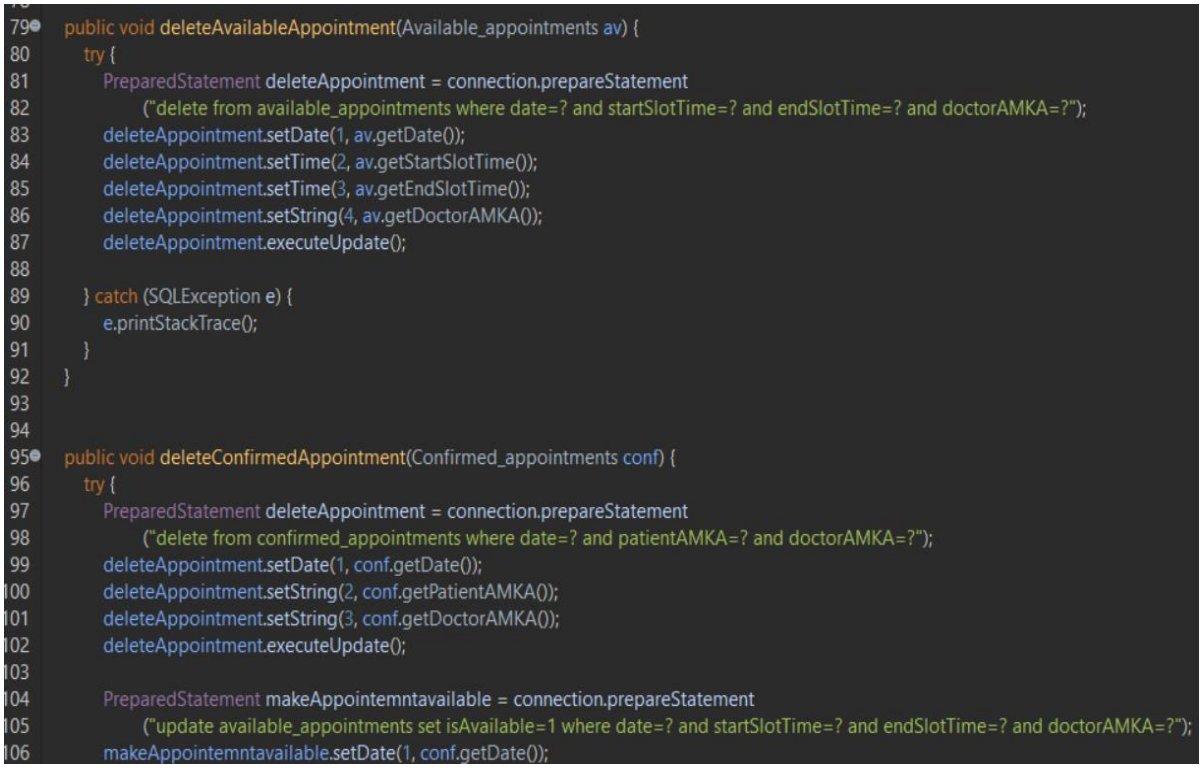
```
286          while(next) {
287
288              if (uname.equals(rs.getString("username")) && (newhashedpass.equals(rs.getString("hashedpassword") ))  ){
289                  found=true;
290                  name=rs.getString("name");
291                  surname=rs.getString("surname");
292                  doctorAMKA=rs.getString("doctorAMKA");
293                  break;
294              }
295              next=rs.next();
296          }
297       }
298      //rs.close();
299      //con.close();
300      }catch (SQLException sqle) {
301          exceptionThrown = true;
302          //sqle.printStackTrace();
303          req.getRequestDispatcher("doctorLogin.html").include(req, res);
304          out.println("<script>" + "alert(\"Database connection problem.\");" +
305                  "</script>");
306
307      }
308
309      if(found){
310          HttpSession session=req.getSession();
311          session.setAttribute("username", uname);
312          session.setAttribute("pass", pass);
313          session.setAttribute("name", name);
```

```
313             session.setAttribute("name", name);
314             session.setAttribute("surname", surname);
315             session.setAttribute("doctorAMKA", doctorAMKA);
316
317
318             res.sendRedirect("doctorProfile.jsp");
319         } else {
320             if(!exceptionThrown) {
321                 req.getRequestDispatcher("doctorLogin.html").include(req, res);
322                 out.println("<script>" + "alert(\"Account wasn't found\");" + "</script>");
323              }
324         }
325
326         out.flush();
327         out.close();
328     }
329
330
331     public static void Logout(HttpServletRequest req,HttpServletResponse res) throws ServletException,IOException{
332
333     HttpSession session=req.getSession();
334     session.removeAttribute("username");
335     session.removeAttribute("pass");
336     session.removeAttribute("name");
337     session.removeAttribute("surname");
338     session.removeAttribute("doctorAMKA");
339     session.invalidate();
340     }
341 }
342
```

6.2.3 Doctor Dao

```
1  package dao;
2
3  import java.io.IOException;
21
22 public class doctorDao {
23    public Connection connection;
24
25    public doctorDao() {
26       connection = DbUtil.getConnection();
27    }
28    private static String LIST_AvailableAppointments = "/ListAppointmentAvailability.jsp";
29    private static String INSERT_AvailableAppointment = "/insertAvailableAppointment.jsp";
30
31    public void insertAvailableAppointment(Available_appointments av_appointment,HttpServletRequest req,HttpServletResponse res)
32          throws IOException, ServletException {
33       PrintWriter out=res.getWriter();
34       HttpSession session=req.getSession();
35       try {
36          PreparedStatement insertappointment = connection.prepareStatement(av_appointment.addAvailableAppointment());
37          insertappointment.setDate(1,av_appointment.getDate());
38          insertappointment.setTime(2, av_appointment.getStartSlotTime());
39          insertappointment.setTime(3,  av_appointment.getEndSlotTime());
40          insertappointment.setString(4, av_appointment.getDoctorAMKA());
41          insertappointment.setInt(5, 1);
42          insertappointment.executeUpdate();
43          req.setAttribute("AvailableAppointments", getAllAvailableAppointmentsByDoctorAMKA(session.getAttribute("doctorAMKA").toString()));
44          req.getRequestDispatcher(LIST_AvailableAppointments).include(req, res);
45          out.println("<script>" + "alert(\"Successful insertion of available appointment\");" +
46                "</script>");
```

```
48       } catch(SQLException sqle ) {
49          //sqle.printStackTrace();
50          req.getRequestDispatcher(INSERT_AvailableAppointment).include(req, res);
51          out.println("<script>" + "alert(\"This available appointment already exists.Please give a new available appointment.\");" +
52                "</script>");
53
54       }
55    }
56    public List<Confirmed_appointments> getAllPendingAppointmentsbyDate(int dateFactor,String doctorAMKA) {
57       List<Confirmed_appointments> PendingAppointments = new ArrayList<Confirmed_appointments>();
58       try {
59          PreparedStatement statement = connection.prepareStatement
60                ("Select * from confirmed_appointments as A natural join patient as P where date >= current_date() "
61                      + "and date <date_add(current_date(), interval ? day) and doctorAMKA=? order by date");
62          statement.setInt(1, dateFactor);
63          statement.setString(2, doctorAMKA);
64          ResultSet rs = statement.executeQuery();
65          while (rs.next()) {
66             Confirmed_appointments Pending_appointment = new
67                   Confirmed_appointments(rs.getDate("date"),rs.getTime("startSlotTime"),rs.getTime("endSlotTime"),
68                         rs.getString("patientAMKA"),rs.getString("doctorAMKA"));
69             Pending_appointment.setPatientNAME(rs.getString("P.name"));
70             PendingAppointments.add(Pending_appointment);
71          }
72       } catch (SQLException e) {
73          e.printStackTrace();
74       }
75
76       return PendingAppointments;
```



```
79  public void deleteAvailableAppointment(Available_appointments av) {
80      try {
81          PreparedStatement deleteAppointment = connection.prepareStatement
82                  ("delete from available_appointments where date=? and startSlotTime=? and endSlotTime=? and doctorAMKA=?");
83          deleteAppointment.setDate(1, av.getDate());
84          deleteAppointment.setTime(2, av.getStartSlotTime());
85          deleteAppointment.setTime(3, av.getEndSlotTime());
86          deleteAppointment.setString(4, av.getDoctorAMKA());
87          deleteAppointment.executeUpdate();
88
89      } catch (SQLException e) {
90          e.printStackTrace();
91      }
92  }
93
94
95  public void deleteConfirmedAppointment(Confirmed_appointments conf) {
96      try {
97          PreparedStatement deleteAppointment = connection.prepareStatement
98                  ("delete from confirmed_appointments where date=? and patientAMKA=? and doctorAMKA=?");
99          deleteAppointment.setDate(1, conf.getDate());
100         deleteAppointment.setString(2, conf.getPatientAMKA());
101         deleteAppointment.setString(3, conf.getDoctorAMKA());
102         deleteAppointment.executeUpdate();
103
104         PreparedStatement makeAppointemntavailable = connection.prepareStatement
105                 ("update available_appointments set isAvailable=1 where date=? and startSlotTime=? and endSlotTime=? and doctorAMKA=?");
106         makeAppointemntavailable.setDate(1, conf.getDate());
```

```
107         makeAppointemntavailable.setTime(2, conf.getStartSlotTime());
108         makeAppointemntavailable.setTime(3, conf.getEndSlotTime());
109         makeAppointemntavailable.setString(4, conf.getDoctorAMKA());
110         makeAppointemntavailable.executeUpdate();
111     } catch (SQLException e) {
112         e.printStackTrace();
113     }
114 }
115
116 public List<Available_appointments> getAllAvailableAppointmentsByDoctorAMKA(String doctorAMKA) {
117     List<Available_appointments> AvailableAppointments = new ArrayList<Available_appointments>();
118     try {
119         PreparedStatement statement = connection.prepareStatement
120                 ("Select * from available_appointments as A natural join doctor as D where date >= now() "
121                         + "and isAvailable=1 and doctorAMKA=? order by date");
122         statement.setString(1,doctorAMKA);
123         ResultSet rs = statement.executeQuery();
124
125         while (rs.next()) {
126             Available_appointments Available_appointment =
127                     new Available_appointments(rs.getDate("date"),rs.getTime("startSlotTime"),rs.getTime("endSlotTime"),rs.getString("doctorAMKA"));
128             Available_appointment.setDoctorNAME(rs.getString("name"));
129             AvailableAppointments.add(Available_appointment);
130         }
131     } catch (SQLException e) {
132         e.printStackTrace();
133     }
134
135     return AvailableAppointments;
136 }
```

```
138  public void updateAvailableAppointment(Available_appointments new_appointment,Available_appointments old_appointment) {
139      try {
140        PreparedStatement updateAvailableAppointment = connection.prepareStatement
141              ("update available_appointments  set date=? ,startSlotTime=? , endSlotTime=? where date=? "
142                  + "and startSlotTime=? and endSlotTime=? and doctorAMKA=?");
143        updateAvailableAppointment.setDate(1, new_appointment.getDate());
144        updateAvailableAppointment.setTime(2, new_appointment.getStartSlotTime());
145        updateAvailableAppointment.setTime(3, new_appointment.getEndSlotTime());
146
147        updateAvailableAppointment.setDate(4, old_appointment.getDate());
148        updateAvailableAppointment.setTime(5, old_appointment.getStartSlotTime());
149        updateAvailableAppointment.setTime(6, old_appointment.getEndSlotTime());
150        updateAvailableAppointment.setString(7, old_appointment.getDoctorAMKA());
151        updateAvailableAppointment.executeUpdate();
152
153      } catch (SQLException e) {
154        e.printStackTrace();
155      }
156
157  }
```

6.3 Κώδικας Admin

6.3.1 Κλάση Admin

```
1  package Classes;
2
3  public class Admin extends  Users {
4      public int userid;
5      public Admin(String username, String password) {
6          super(username,password);
7          this.userid=Users.getUsersId();
8      }
9
10     public Doctor createDoctor(String username, String password, String name, String surname,String specialty,String amka){
11
12         Doctor doctor=new Doctor(username, password, name, surname, specialty,amka);
13         return doctor;
14     }
15     public String registration(){
16         String insertAdminStatement = "INSERT INTO ADMIN ( username, hashedpassword, salt) VALUES (?,?,?)";
17         return(insertAdminStatement);
18     }
19 }
20
```

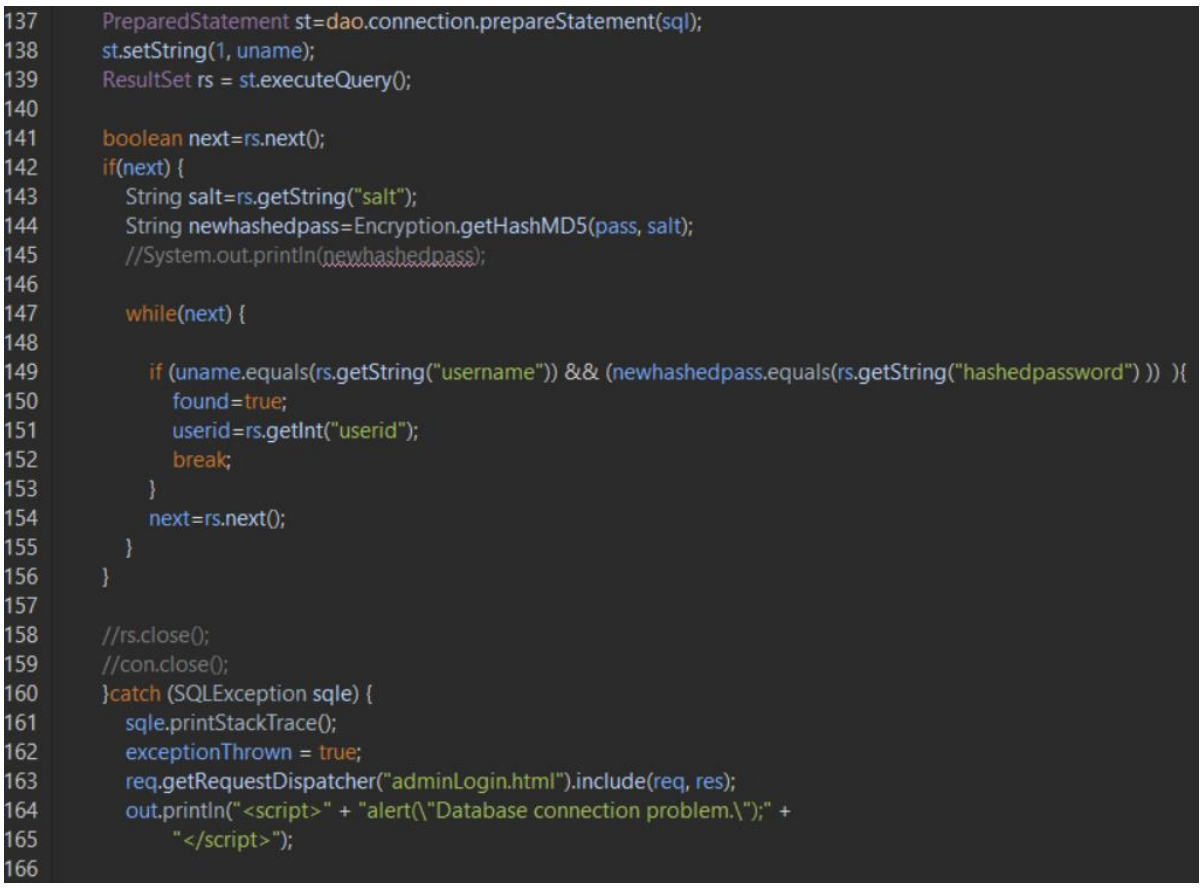
6.3.2 Admin Servlet

```
1  package Servlets;
2
3  import java.io.IOException;
19
20 @WebServlet("/AdminServlet")
21 public class AdminServlet extends HttpServlet {
22
23    private static final long serialVersionUID = 1L;
24
25    private static String INSERT_Doctor = "/addDoctor.jsp";
26    private static String EDIT_Doctor = "/editDoctor.jsp";
27    private static String LIST_Doctor = "/listDoctor.jsp";
28
29    private static adminDao dao;
30
31    public AdminServlet() {
32       super();
33       dao = new adminDao();
34    }
35
36    public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
37       String forward="";
38       String action = request.getParameter("action");
39       response.setContentType("text/html; charset=UTF-8");
40       response.setCharacterEncoding("UTF-8");
41       request.setCharacterEncoding("UTF-8");
42
43       if (action.equalsIgnoreCase("delete")){
44          String doctorAMKA = request.getParameter("doctorAMKA");
45          dao.deleteDoctor(doctorAMKA);
```

```
46          forward = LIST_Doctor;
47          request.setAttribute("Doctors", dao.getAllDoctors());
48       } else if (action.equalsIgnoreCase("edit")){
49          forward = EDIT_Doctor;
50          String doctorAMKA = request.getParameter("doctorAMKA");
51          Doctor doctor = dao.getDoctorBydoctorAMKA(doctorAMKA);
52          request.setAttribute("Doctor", doctor);
53
54       } else if (action.equalsIgnoreCase("listDoctor")){
55          forward = LIST_Doctor;
56          request.setAttribute("Doctors", dao.getAllDoctors());
57       } else {
58          forward = INSERT_Doctor;
59       }
60
61       RequestDispatcher view = request.getRequestDispatcher(forward); //σε παει στην αλλη σελιδα
62       view.forward(request, response);
63    }
64    public void doPost(HttpServletRequest req,HttpServletResponse res) throws ServletException,IOException{
65       PrintWriter out=res.getWriter();
66       res.setContentType("text/html; charset=UTF-8");
67       res.setCharacterEncoding("UTF-8");
68       req.setCharacterEncoding("UTF-8");
69
70       HttpSession session=req.getSession();
71       if(session.getAttribute("doctorAMKA")!=null) {
72          DoctorServlet.Logout(req, res);
73       }else if(session.getAttribute("patientAMKA")!=null) {
74          PatientServlet.Logout(req, res);
75       }
```

```
77       if(req.getParameter("LoginButton")!=null){
78          Login(req,res);
79
80       }else if(req.getParameter("SignupButton")!=null){
81          SignupAdmin(req,res);
82       }
83       else if(req.getParameter("LogoutButton")!=null) {
84          Logout(req,res);
85          res.sendRedirect("adminLogin.html");
86       } else if (req.getParameter("edit")!=null){
87          boolean temp = false;
88          String doctorAMKA = req.getParameter("doctorAMKA");
89          String username = req.getParameter("username");
90          String hashedpassword=req.getParameter("hashedpassword");
91          String password = req.getParameter("newpassword");
92          String name = req.getParameter("name");
93          String surname = req.getParameter("surname");
94          String specialty = req.getParameter("specialty");
95
96          if (!isNullOrBlankOrWhiteSpace(name)  && !isNullOrBlankOrWhiteSpace(surname) ){
97
98             if(isNullOrBlankOrWhiteSpace(password)) {
99                Doctor doctor = new Doctor(username,hashedpassword,name,surname,specialty,doctorAMKA);
00                dao.updateDoctor(doctor);
01             }else{
02                String salt=Encryption.createSalt().toString();
03                Doctor doctor = new Doctor(username,Encryption.getHashMD5(password, salt),name,surname,specialty,doctorAMKA);
04                doctor.setSalt(salt);
05                dao.updateDoctor(doctor);
```

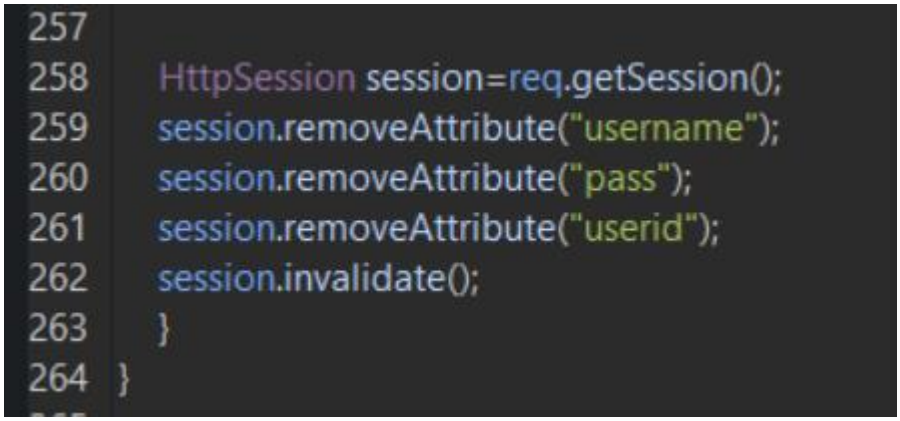
```
107         }else {
108             temp = true;
109             Doctor doctor = dao.getDoctorBydoctorAMKA(doctorAMKA);
110             req.setAttribute("Doctor", doctor);
111             req.getRequestDispatcher(EDIT_Doctor).include(req, res);
112             out.println("<script>" + "alert(\"Inputs  cannot be blank or contain white spaces\");" +
113                     "</script>");
114          }
115
116         if(!temp) {
117             RequestDispatcher view = req.getRequestDispatcher(LIST_Doctor);
118             req.setAttribute("Doctors", dao.getAllDoctors());
119             view.forward(req, res);
120         }
121
122     }else if(req.getParameter("insert")!=null) {
123         addDoctor(req,res);
124     }
125 }
126 public void Login(HttpServletRequest req,HttpServletResponse res) throws IOException, ServletException{
127     PrintWriter out=res.getWriter();
128     String uname=req.getParameter("username");
129     String pass=req.getParameter("pass");
130     int userid = 0;
131
132     String sql="Select * from admin where username=?";
133     boolean found=false;
134     boolean exceptionThrown = false;
135     try {
136
```



```
137     PreparedStatement st=dao.connection.prepareStatement(sql);
138     st.setString(1, uname);
139     ResultSet rs = st.executeQuery();
140
141     boolean next=rs.next();
142     if(next) {
143        String salt=rs.getString("salt");
144        String newhashedpass=Encryption.getHashMD5(pass, salt);
145        //System.out.println(newhashedpass);
146
147        while(next) {
148
149           if (uname.equals(rs.getString("username")) && (newhashedpass.equals(rs.getString("hashedpassword") )) ){
150              found=true;
151              userid=rs.getInt("userid");
152              break;
153           }
154           next=rs.next();
155        }
156     }
157
158     //rs.close();
159     //con.close();
160     }catch (SQLException sqle) {
161        sqle.printStackTrace();
162        exceptionThrown = true;
163        req.getRequestDispatcher("adminLogin.html").include(req, res);
164        out.println("<script>" + "alert(\"Database connection problem.\");" +
165              "</script>");
166
```

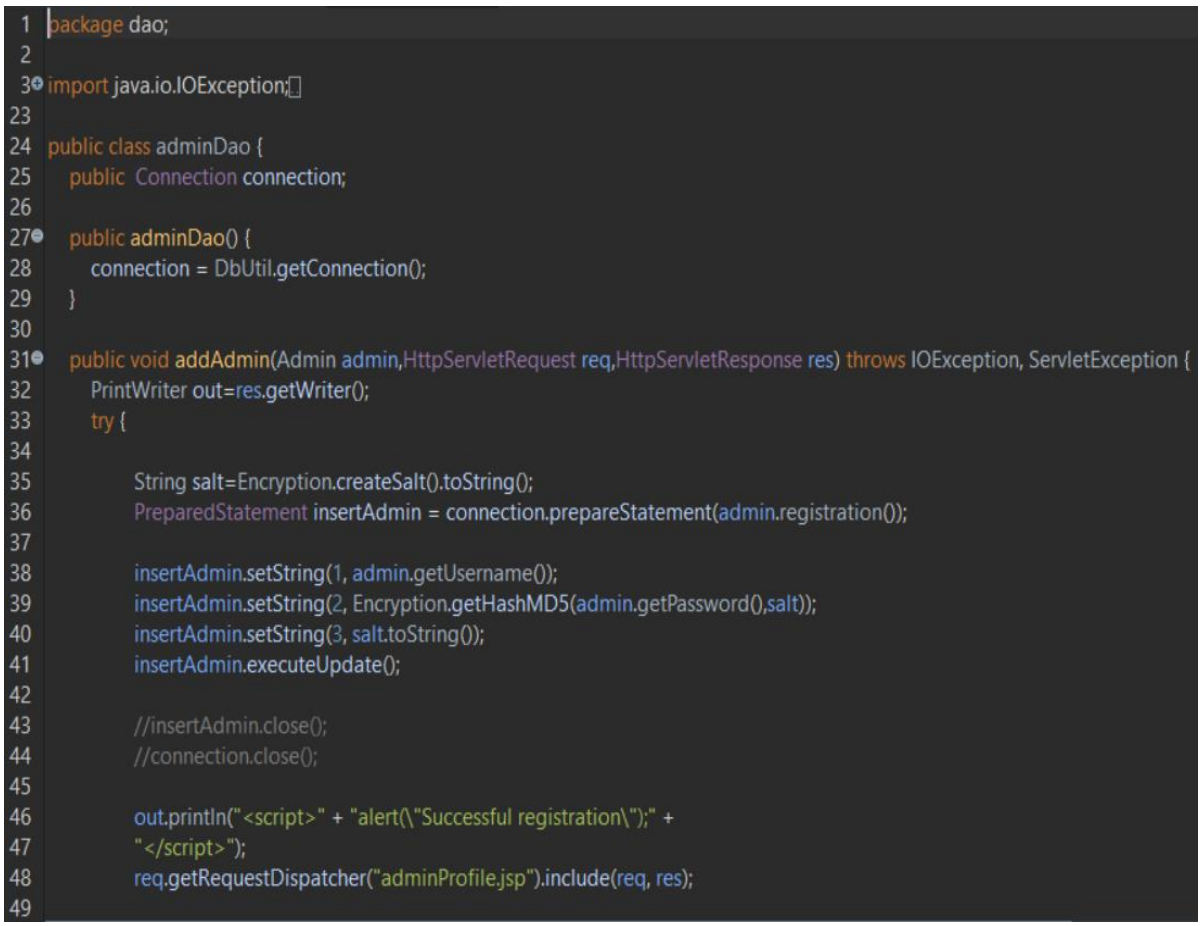
```
169     if(found){
170        HttpSession session=req.getSession();
171        session.setAttribute("username", uname);
172        session.setAttribute("pass", pass);
173        session.setAttribute("userid", userid);
174
175        res.sendRedirect("adminProfile.jsp");
176     }else {
177        if(!exceptionThrown) {
178
179           req.getRequestDispatcher("adminLogin.html").include(req, res);
180           out.println("<script>" + "alert(\"Account wasn't found\");" +
181                 "</script>");
182        }
183     }
184     out.flush();
185     out.close();
186  }
187
188
189
190
191  public void SignupAdmin(HttpServletRequest req,HttpServletResponse res) throws IOException, ServletException{
192     PrintWriter out=res.getWriter();
193
194     String username = req.getParameter("username");
195     String password = req.getParameter("pass");
196
197     if(!isNullOrBlankOrWhiteSpace(username)&&!isNullOrBlankOrWhiteSpace(password)){
198        Admin admin = new Admin(username,password);
```

```
199         dao.addAdmin(admin,req,res);
200
201     }else {
202
203         req.getRequestDispatcher("SignUpAdmin.jsp").include(req, res);
204         out.println("<script>" + "alert(\"Inputs  cannot be blank or contain white spaces\");" +
205                 "</script>");
206
207     }
208
209   out.flush();
210   out.close();
211   }
212
213   public void addDoctor(HttpServletRequest req,HttpServletResponse res) throws IOException, ServletException{
214       PrintWriter out=res.getWriter();
215       String doctorAMKA = req.getParameter("AMKA");
216       String username = req.getParameter("username");
217       String password = req.getParameter("pass");
218       String name = req.getParameter("firstname");
219       String surname = req.getParameter("lastname");
220       String specialty = req.getParameter("specialty");
221       if(!isNullOrBlankOrWhiteSpace(doctorAMKA)&&!isNullOrBlankOrWhiteSpace(username)
222               &&!isNullOrBlankOrWhiteSpace(password)&&!isNullOrBlankOrWhiteSpace(name)&&!isNullOrBlankOrWhiteSpace(surname)){
223           Doctor doctor = new Doctor(username,password,name,surname,specialty,doctorAMKA);
224           dao.addDoctor(doctor, req, res);
225
226       }else {
227
228           req.getRequestDispatcher(INSERT_Doctor).include(req, res);
229           out.println("<script>" + "alert(\"Inputs  cannot be blank or contain white spaces\");" +
230                   "</script>");
231
232       }
233
234       out.flush();
235       out.close();
236   }
237
238   public static boolean isNullOrBlankOrWhiteSpace(String value) {
239
240       if (value == null) {
241           return true;
242       }
243       if (value.length() == 0) {
244           return true;
245       }
246       for (int i = 0; i < value.length(); i++) {
247           if (Character.isWhitespace(value.charAt(i))) {
248               return true;
249           }
250       }
251
252       return false;
253   }
254
255
256   public static void Logout(HttpServletRequest req,HttpServletResponse res) throws ServletException,IOException{
257
```

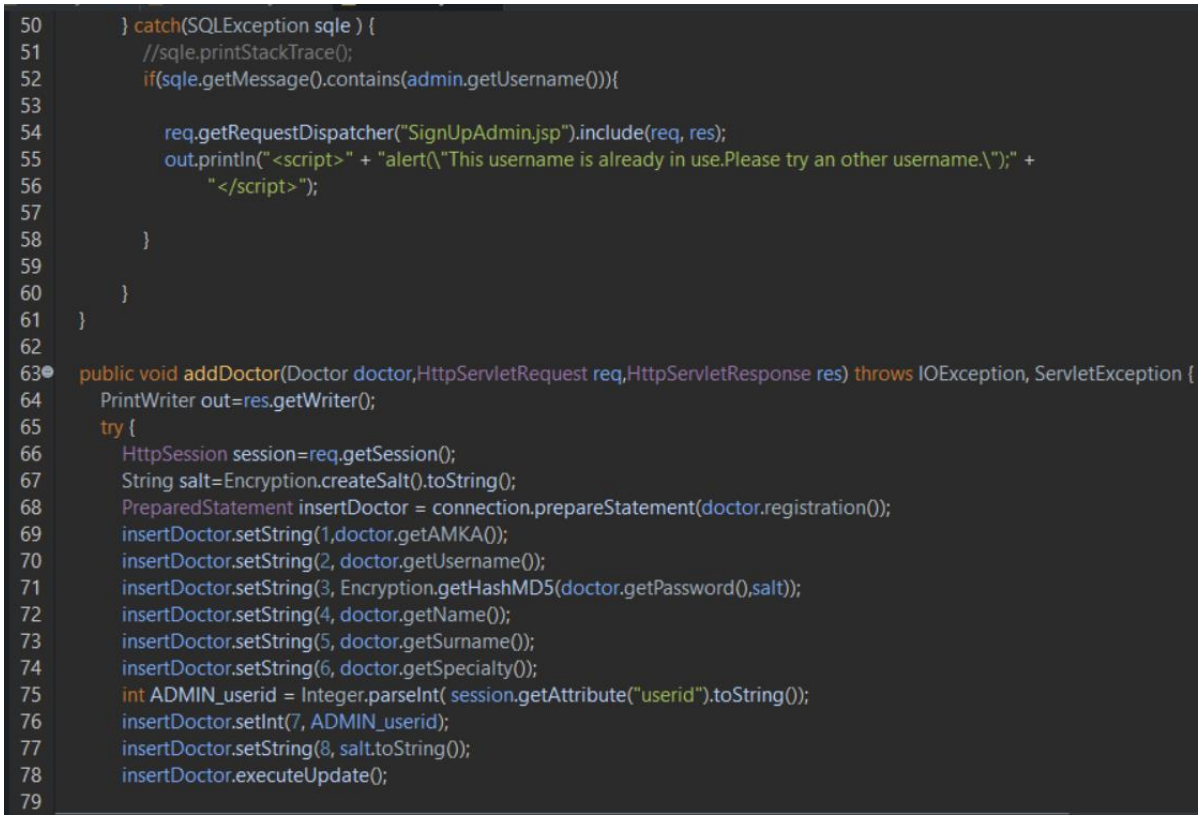


```
257
258     HttpSession session=req.getSession();
259     session.removeAttribute("username");
260     session.removeAttribute("pass");
261     session.removeAttribute("userid");
262     session.invalidate();
263     }
264 }
```

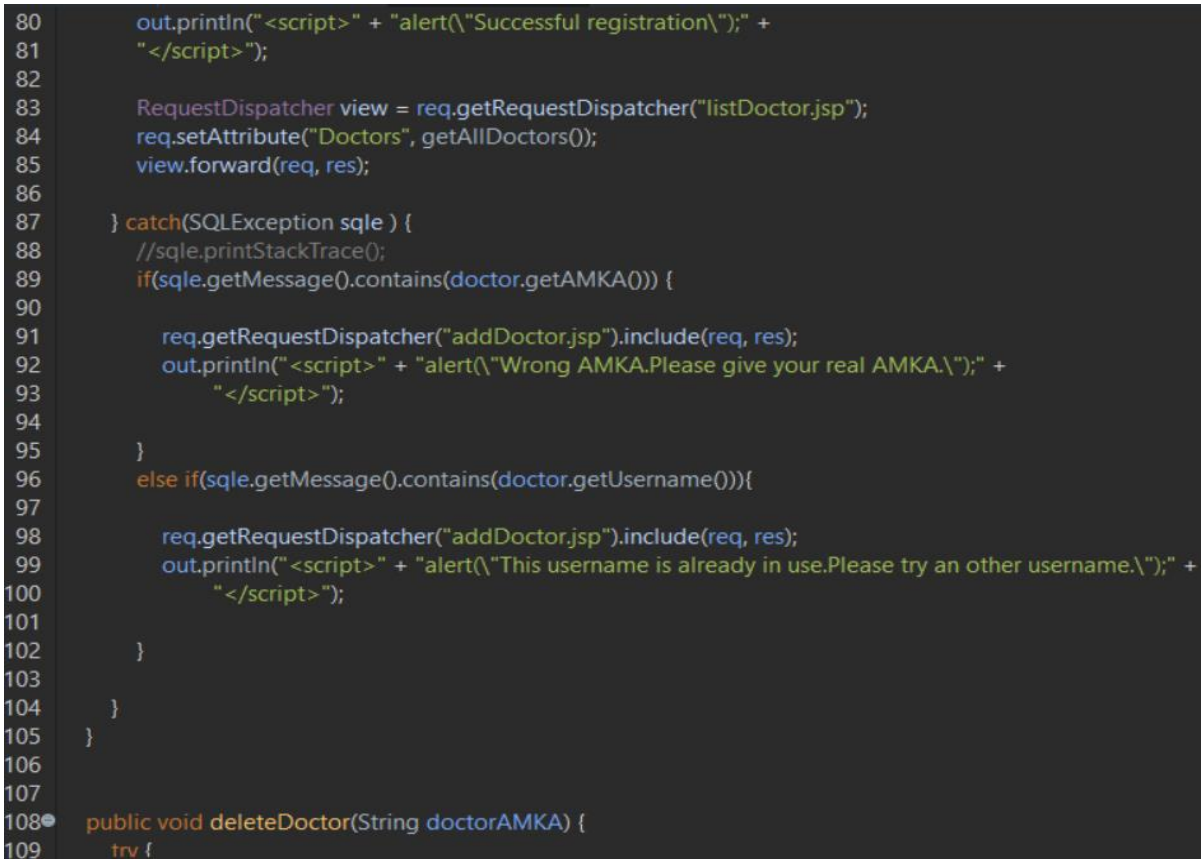
6.3.3 Admin Dao



```
1  package dao;
2
3  import java.io.IOException;
23
24 public class adminDao {
25    public  Connection connection;
26
27    public adminDao() {
28       connection = DbUtil.getConnection();
29    }
30
31    public void addAdmin(Admin admin,HttpServletRequest req,HttpServletResponse res) throws IOException, ServletException {
32       PrintWriter out=res.getWriter();
33       try {
34
35             String salt=Encryption.createSalt().toString();
36             PreparedStatement insertAdmin = connection.prepareStatement(admin.registration());
37
38             insertAdmin.setString(1, admin.getUsername());
39             insertAdmin.setString(2, Encryption.getHashMD5(admin.getPassword(),salt));
40             insertAdmin.setString(3, salt.toString());
41             insertAdmin.executeUpdate();
42
43             //insertAdmin.close();
44             //connection.close();
45
46             out.println("<script>" + "alert(\"Successful registration\");" +
47             "</script>");
48             req.getRequestDispatcher("adminProfile.jsp").include(req, res);
49
```



```
50        } catch(SQLException sqle ) {
51           //sqle.printStackTrace();
52           if(sqle.getMessage().contains(admin.getUsername())){
53
54              req.getRequestDispatcher("SignUpAdmin.jsp").include(req, res);
55              out.println("<script>" + "alert(\"This username is already in use.Please try an other username.\");" +
56                    "</script>");
57
58           }
59
60        }
61     }
62
63     public void addDoctor(Doctor doctor,HttpServletRequest req,HttpServletResponse res) throws IOException, ServletException {
64        PrintWriter out=res.getWriter();
65        try {
66           HttpSession session=req.getSession();
67           String salt=Encryption.createSalt().toString();
68           PreparedStatement insertDoctor = connection.prepareStatement(doctor.registration());
69           insertDoctor.setString(1,doctor.getAMKA());
70           insertDoctor.setString(2, doctor.getUsername());
71           insertDoctor.setString(3, Encryption.getHashMD5(doctor.getPassword(),salt));
72           insertDoctor.setString(4, doctor.getName());
73           insertDoctor.setString(5, doctor.getSurname());
74           insertDoctor.setString(6, doctor.getSpecialty());
75           int ADMIN_userid = Integer.parseInt( session.getAttribute("userid").toString());
76           insertDoctor.setInt(7, ADMIN_userid);
77           insertDoctor.setString(8, salt.toString());
78           insertDoctor.executeUpdate();
79
```

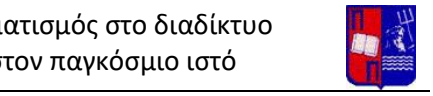


```
80           out.println("<script>" + "alert(\"Successful registration\");" +
81           "</script>");
82
83           RequestDispatcher view = req.getRequestDispatcher("listDoctor.jsp");
84           req.setAttribute("Doctors", getAllDoctors());
85           view.forward(req, res);
86
87        } catch(SQLException sqle ) {
88           //sqle.printStackTrace();
89           if(sqle.getMessage().contains(doctor.getAMKA())) {
90
91              req.getRequestDispatcher("addDoctor.jsp").include(req, res);
92              out.println("<script>" + "alert(\"Wrong AMKA.Please give your real AMKA.\");" +
93                    "</script>");
94
95           }
96           else if(sqle.getMessage().contains(doctor.getUsername())){
97
98              req.getRequestDispatcher("addDoctor.jsp").include(req, res);
99              out.println("<script>" + "alert(\"This username is already in use.Please try an other username.\");" +
100                   "</script>");
101
102          }
103
104       }
105    }
106
107
108    public void deleteDoctor(String doctorAMKA) {
109       try {
```

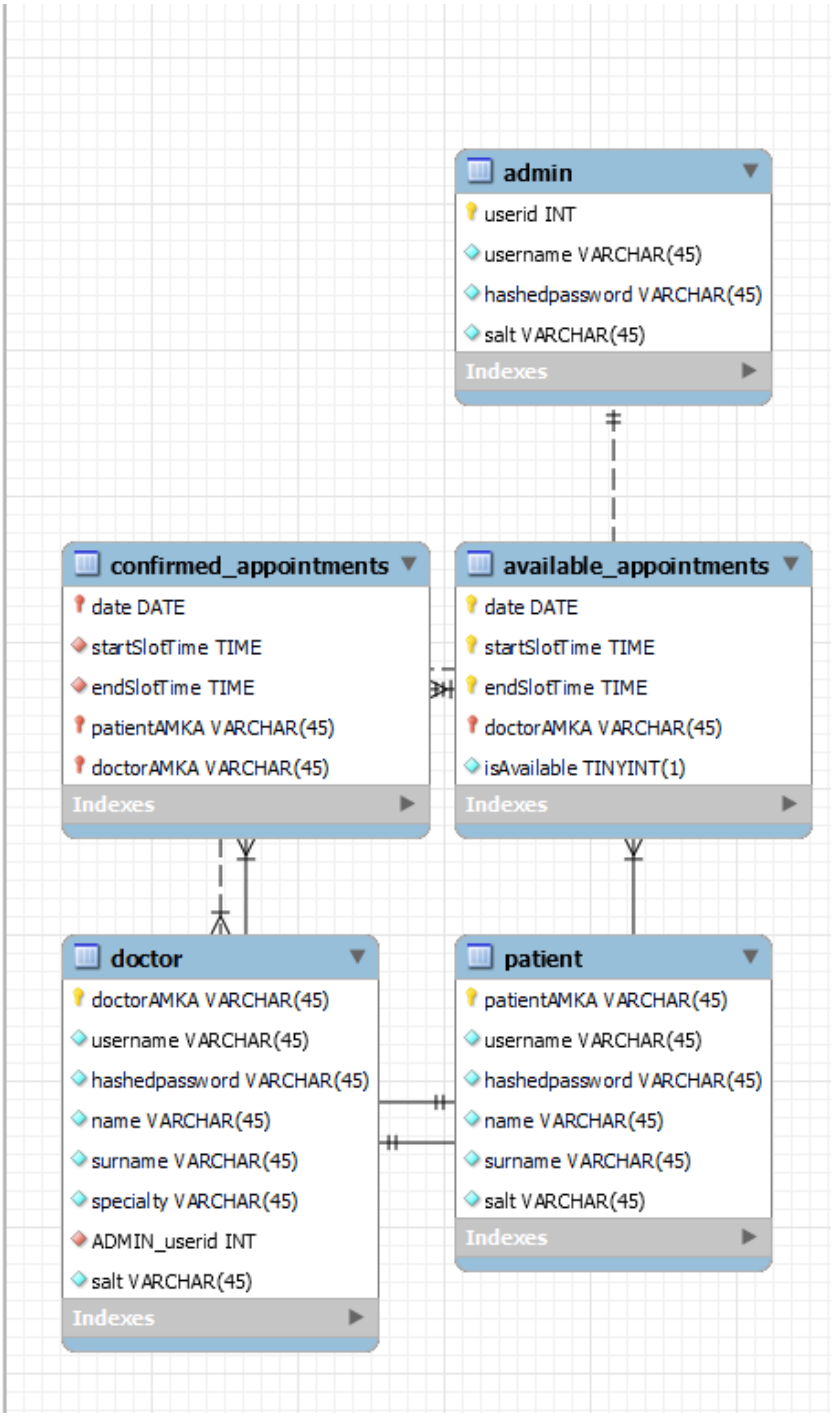
```
109     try {
110       PreparedStatement deleteDoctor = connection.prepareStatement("delete from doctor where doctorAMKA=?");
111       deleteDoctor.setString(1, doctorAMKA);
112       deleteDoctor.executeUpdate();
113
114     } catch (SQLException e) {
115       e.printStackTrace();
116     }
117   }
118
119   public void updateDoctor(Doctor doctor) {
120     try {
121       if(doctor.getSalt()==null) {
122         PreparedStatement updateDoctor= connection.prepareStatement
123             ("update doctor set  username=?, hashedPassword=?, name=?, surname=?, specialty=? where doctorAMKA=?");
124
125         updateDoctor.setString(1, doctor.getUsername());
126         updateDoctor.setString(2, doctor.getPassword());
127         updateDoctor.setString(3, doctor.getName());
128         updateDoctor.setString(4, doctor.getSurname());
129         updateDoctor.setString(5, doctor.getSpecialty());
130         updateDoctor.setString(6, doctor.getAMKA());
131         updateDoctor.executeUpdate();
132         updateDoctor.close();
133
134       }else {
135         PreparedStatement updateDoctor= connection.prepareStatement
136             ("update doctor set  username=?, hashedPassword=?, name=?, surname=?, specialty=?, salt=? where doctorAMKA=?");
137
```

```
138         updateDoctor.setString(1, doctor.getUsername());
139         updateDoctor.setString(2, doctor.getPassword());
140         updateDoctor.setString(3, doctor.getName());
141         updateDoctor.setString(4, doctor.getSurname());
142         updateDoctor.setString(5, doctor.getSpecialty());
143         updateDoctor.setString(6, doctor.getSalt());
144         updateDoctor.setString(7, doctor.getAMKA());
145         updateDoctor.executeUpdate();
146       }
147     } catch (SQLException e) {
148       e.printStackTrace();
149     }
150
151   }
152
153   public List<Doctor> getAllDoctors() {
154     List<Doctor> Doctors = new ArrayList<Doctor>();
155     try {
156       Statement statement = connection.createStatement();
157       ResultSet rs = statement.executeQuery("select * from doctor");
158       while (rs.next()) {
159         Doctor doctor = new Doctor(rs.getString("username"),
160             rs.getString("hashedpassword"),rs.getString("name"),rs.getString("surname"),rs.getString("specialty"),rs.getString("doctorAMKA"));
161         Doctors.add(doctor);
162       }
163     } catch (SQLException e) {
164       e.printStackTrace();
165     }
166
167     return Doctors;
```

```
169
170    public Doctor getDoctorBydoctorAMKA(String doctorAMKA) {
171        Doctor doctor = null;
172        try {
173            PreparedStatement preparedStatement = connection.prepareStatement("select * from Doctor where doctorAMKA=?");
174            preparedStatement.setString(1, doctorAMKA);
175            ResultSet rs = preparedStatement.executeQuery();
176
177            if (rs.next()) {
178                doctor = new Doctor(rs.getString("username"),
179                        rs.getString("hashedpassword"),rs.getString("name"),rs.getString("surname"),rs.getString("specialty"),rs.getString("doctorAMKA"));
180            }
181        } catch (SQLException e) {
182            e.printStackTrace();
183        }
184
185        return doctor;
186    }
187 }
188
```



7 Βάση Δεδομένων



Η ιδέα πίσω από την βάση είναι ότι στο **available_appointments** κατοχυρώνονται τα διαθέσιμα ραντεβού προς τους **ασθενείς** απτον κάθε **Ιατρό**

Αφότου επιλέξει ο ασθενής και το κλείσει τότε στο συγκεκριμενο ραντεβού θετουμε το isAvailable = 0(false) και το μεταφέρουμε στον πίνακα **confirmed appointments**

7.1 Κώδικας σύνδεσης Βάσης

```
1  package util;
2
3  import java.sql.Connection;
8
9  public class DbUtil {
10
11     private static Connection connection = null;
12     private static DataSource datasource = null;
13
14     public static Connection getConnection() {
15        if (connection != null)
16           return connection;
17        else {
18           try {
19              InitialContext ctx = new InitialContext();
20              datasource = (DataSource)ctx.lookup("java:comp/env/jdbc/ProjectDataSource_2");
21              connection = datasource.getConnection();
22           }catch(Exception e) {
23              e.printStackTrace();
24           }
25           return connection;
26
27        }
28
29     }
30 }
```

8 Μηχανισμός Εφαρμογής(Encryption)

```
1  package Classes;
2  import java.math.BigInteger;
6
7  public class Encryption {
8
9      public static String getHashMD5(String unhashed, String salt) {
10         // Hash the password.
11         final String toHash = salt + unhashed + salt;
12         MessageDigest messageDigest = null;
13         try {
14             messageDigest = MessageDigest.getInstance("MD5");
15         } catch (NoSuchAlgorithmException ex) {
16             return "00000000000000000000000000000000";
17         }
18         messageDigest.reset();
19         messageDigest.update(toHash.getBytes(), 0, toHash.length());
20         String hashed = new BigInteger(1, messageDigest.digest()).toString(16);
21         if (hashed.length() < 32) {
22             hashed = "0" + hashed;
23         }
24         return hashed.toUpperCase();
25     }
26
27 public static byte[] createSalt(){
28     byte[] bytes =new byte[20];
29     SecureRandom random=new SecureRandom();
30     random.nextBytes(bytes);
31     return bytes;
32 }
```

Το password κάθε Χρήστη της εφαρμογής αποθηκεύεται αλλά και ταυτοποιείτε σε hashed μορφή από τον αλγόριθμο MD5

Επίσης χρησιμοποιείτε η παραμετροποίηση τύπου prepared statements για να εκτελεστούν τα ερωτήματα προς την βάση ώστε η εφαρμογή να είναι ασφαλής από επιθέσεις όπως SQL injection και ο κώδικας πιο ευέλικτος και βέλτιστος.

9 Ενδεικτικά JSP/HTML αρχεία

- Σε ολες τις jsp σελιδες που εμφανιζεται ενας πινακας με στοιχεια που παιρνουμε μεσω της βασης δεδομενων δεχονται μια λιστα μεσω του request η οποια στελνεται από το servlet,δημιουργουν τον πινακα με τις αντιστοιχες στηλες και στη συνεχεια με ένα for each loop παιρνουμε ένα ένα τα στοιχεια της λιστας και τα περναμε στα κελια του πινακα.

- Αντίστοιχα οι jsp σελίδες που κάνουν κάποιο edit δέχονται το συγκεκριμένο αντικείμενο το οποίο στέλνεται από το servlet και ο χρήστης θέλει να κάνει update ώστε να βλέπει τα χαρακτηριστικά που μπορεί να αλλάξει.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>admin login</title>

  <link rel="stylesheet" type="text/css" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/4.0.0-beta/css/bootstrap.m
  <link rel="stylesheet" type="text/css" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.5.0/css/font-awesome.min.css"
  <link rel="stylesheet" type="text/css" href="https://fonts.googleapis.com/css?family=Roboto">
  <link rel="stylesheet" type="text/css" href="css/admin.css">
</head>
<body >
   <ul>
  <li><a href="index.html">Home</a></li>
  <li><a href="patientLogin.html">Patient</a></li>
  <li><a href="doctorLogin.html">Doctor</a></li>
  <li><a class="active" href="adminLogin.html">Admin</a></li>
  <li style="float:right"><a href="about.html">About</a></li>
</ul>
    <div class="container">
        <div class="row">
            <div class="col-lg-3 col-md-2"></div>
            <div class="col-lg-6 col-md-8 login-box">
                <div class="col-lg-12 login-key">
                    <i class="fa fa-key" aria-hidden="true"></i>
                </div>
                <div class="col-lg-12 login-title">
                    ADMIN PANEL
                </div>

                <div class="col-lg-12 login-form">
                    <div class="col-lg-12 login-form">
                        <form action="AdminServlet" method="post">
                            <div class="form-group">
                                <label class="form-control-label">USERNAME</label>
                                <input type="text" class="form-control"  name="username" required autocomplete="off">
```

```
                                <input type="text" class="form-control"  name="username" required autocomplete="off">
                            </div>
                            <div class="form-group">
                                <label class="form-control-label">PASSWORD</label>
                                <input type="password" class="form-control"  name="pass" required>
                            </div>

                            <div class="col-lg-12 loginbttm">
                                <div class="col-lg-6 login-btm login-text">
                                    <!-- Error Message -->
                                </div>
                                <div class="col-lg-6 login-btm login-button">
                                    <button type="submit" class="btn btn-outline-primary" name="LoginButton">LOGIN</button>
                                </div>
                            </div>
                        </form>
                    </div>
                </div>
                <div class="col-lg-3 col-md-2"></div>
            </div>
        </div>
      </div>

</body>
</html>
```

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>SignupAdmin</title>
<link rel="stylesheet" type="text/css" href="css/main.css">
</head>
<body>
   <%
response.setHeader("Cache-Control","no-cache,no-store,must_revalidate");

if(session.getAttribute("userid")==null){
    response.sendRedirect("adminLogin.html");
}
%>

<div class="limiter">
        <div class="container-login100">
            <div class="wrap-login100">
                <div class="login100-form-title" style="background-image: url(images/signup.jpg); height: auto; width:auto;backgro
                    <span class="login100-form-title-1">
                        Sign Up
                    </span>
                </div>

                <form class="login100-form validate-form" action="AdminServlet" method="post" >
                    <div class="wrap-input100 validate-input m-b-26">
                        <span class="label-input100">Username</span>
                        <input class="input100" type="text" name="username" placeholder="Enter username" required>
                        <span class="focus-input100"></span>
                    </div>

                    <div class="wrap-input100 validate-input m-b-18">
                        <span class="label-input100">Password</span>
```

```
36                      <span class="label-input100">Password</span>
37                      <input class="input100" type="password" name="pass" placeholder="Enter password" required>
38                      <span class="focus-input100"></span>
39                  </div>
40                  <div class="container-login100-form-btn ">
41
42                      <button class="login100-form-btn" name="SignupButton">
43                          Sign Up
44                      </button>
45                  </div>
46
47                  <br><br><br>
48
49              </form>
50
51          </div>
52      </div>
53  </div>
54 </body>
55 </html>
```

```jsp
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1" import ="java.sql.*,Servlets.AdminServlet"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Admin Profile</title>
<link rel="stylesheet" type="text/css" href="css/AdminProfile.css">
</head>
<body style="background-color:#bdd0d6;">
<%
response.setHeader("Cache-Control","no-cache,no-store,must_revalidate");

if(session.getAttribute("userid")==null){
    response.sendRedirect("adminLogin.html");
}
%>

<h1 style="text-align:center;font-size:35px">Admin's Profile Card</h1>

<div class="card">
<form action="AdminServlet" method="post">
<button name="LogoutButton" type="submit" value="Logout" style="position:relative;top:10px;left:330px;width:120px;color:white;back
</form>

  <img src="images/admin.png" alt="admin" style="width:300px; height:180px;">
  <br><br><br><br>
  <p>Username: <%= session.getAttribute("username")%></p>


  <p><button style="background-color: #2B547E;"onclick="javascript:window.location.href='AdminServlet?action=listDoctor';">Registe
  <br>
  <h3 style="text-align: right; "> <strong>  <a href="SignUpAdmin.jsp"><span>  Create a new admin   </span></a></strong> </h3>

</div>

```

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<link REL=STYLESHEET
      HREF="css/JSP-Styles.css"
      TYPE="text/css">
<title>Show All Doctors</title>
</head>
<body>
    <%
response.setHeader("Cache-Control","no-cache,no-store,must_revalidate");

if(session.getAttribute("userid")==null){
    response.sendRedirect("adminLogin.html");
}
%>
 <div>
     <h2>All Doctors</h2>
 <input type="button" onclick="location.href='adminProfile.jsp';" value="Back to Profile" class="myButton" style="float:right;top:
</div>

    <table border=1>
        <thead>
            <tr>
                <th>Doctor's AMKA</th>
                <th>Username</th>
                <th>Name</th>
                <th>SurName</th>
                <th>Specialty</th>
                <th colspan=2>Action</th>
            </tr>
        </thead>
        <tbody>
            <c:forEach items="${requestScope.Doctors}" var="d">
                <tr>
                    <td><c:out value="${d.getAMKA()}" /></td>
                    <td><c:out value="${d.getUsername()}" /></td>
                    <td><c:out value="${d.getName()}" /></td>
                    <td><c:out value="${d.getSurname()}" /></td>
                    <td><c:out value="${d.getSpecialty()}" /></td>
                    <td><a href="AdminServlet?action=edit&doctorAMKA=<c:out value="${d.getAMKA()}"/>">Update</a></td>
                    <td><a href="AdminServlet?action=delete&doctorAMKA=<c:out value="${d.getAMKA()}"/>">Delete</a></td>
                </tr>
            </c:forEach>
        </tbody>
    </table>
    <p><a href="AdminServlet?action=insert" >Add Doctor</a></p>
</body>
</html>
```

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<link REL=STYLESHEET
      HREF="css/JSP-Styles.css"
      TYPE="text/css">
<script type="text/javascript" src="js/jquery-1.7.1.min.js"></script>
<script type="text/javascript" src="js/jquery-ui-1.8.18.custom.min.js"></script>
<title>Edit Doctor</title>
</head>
<body>


<%
response.setHeader("Cache-Control","no-cache,no-store,must_revalidate");

if(session.getAttribute("userid")==null){
    response.sendRedirect("adminLogin.html");
}
%>
<div>
     <h2>Edit Doctor </h2>
 <input type="button" onclick="location.href='adminProfile.jsp';" value="Back to Profile" class="myButton" style="float:right;top:
</div>


    <form method="POST" action='AdminServlet' name="frmeditDoctor">
        AMKA : <input type="text" readonly="readonly"  name="doctorAMKA" value="<c:out value="${Doctor.getAMKA()}" />" /> <br />
        Username : <input type="text" readonly="readonly" name="username" value="<c:out value="${Doctor.getUsername()}" />" /> <br
         <input type="hidden" name="hashedpassword" value="<c:out value="${Doctor.getPassword()}" />" /> <br />
         Password: <input type="password" name="newpassword" value="" /> <br />
         Name : <input type="text" name="name" value="<c:out value="${Doctor.getName()}" />" /> <br />
         Surname : <input type="text" name="surname" value="<c:out value="${Doctor.getSurname()}" />" /> <br />
         <label for="specialty">Choose a Specialty: </label>
     <select name="specialty" >
     <option selected="selected">${Doctor.getSpecialty()}</option>
     <option >Pathologist</option>
     <option >Ophthalmologist</option>
      <option >Orthopedic</option>
     </select>
  <input type="submit" value="Submit" name = "edit" />
    </form>
</body>
</html>
```

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Add Doctor</title>
<link rel="stylesheet" type="text/css" href="css/main.css">
</head>
<body>
<%
response.setHeader("Cache-Control","no-cache,no-store,must_revalidate");

if(session.getAttribute("userid")==null){
    response.sendRedirect("adminLogin.html");
}
%>

<div class="limiter">
        <div class="container-login100">
            <div class="wrap-login100">
                <div class="login100-form-title" style="background-image: url(images/signup.jpg); height: auto; width:auto;backgro
                    <span class="login100-form-title-1">
                        Add Doctor
                    </span>
                </div>

                <form class="login100-form validate-form" action='AdminServlet' name="frmeditDoctor" method="post" >
                    <div class="wrap-input100 validate-input m-b-26">
                        <span class="label-input100">Username</span>
                        <input class="input100" type="text" name="username" placeholder="Enter username" required>
                        <span class="focus-input100"></span>
                    </div>

                    <div class="wrap-input100 validate-input m-b-18">
                        <span class="label-input100">Password</span>
                        <input class="input100" type="password" name="pass" placeholder="Enter password" required>
                        <span class="focus-input100"></span>
```

```
                    </div>
                    <div class="wrap-input100 validate-input m-b-18">
                        <span class="label-input100">First name</span>
                        <input class="input100" type="text" name="firstname" placeholder="Enter First name" required>
                        <span class="focus-input100"></span>
                    </div>
                    <div class="wrap-input100 validate-input m-b-18">
                        <span class="label-input100">Last name</span>
                        <input class="input100" type="text" name="lastname" placeholder="Enter Last name" required>
                        <span class="focus-input100"></span>
                    </div>
                    <div class="wrap-input100 validate-input m-b-18">
                        <span class="label-input100">AMKA</span>
                        <input class="input100" type="text" name="AMKA" placeholder="Enter AMKA" required maxlength="11">
                        <span class="focus-input100"></span>
                    </div>

                    <div class="wrap-input100_2  m-b-18">

                        <span class="label-input100"><br>Specialty</span>
                        <br>
                         <select name="specialty" >
                         <option >Pathologist</option>
                         <option >Ophthalmologist</option>
                         <option >Orthopedic</option>
                         </select>
                        <span class="focus-input100"></span>
                    </div>
                    <br><br><br>
                    <div class="container-login100-form-btn ">

                        <button class="login100-form-btn" name="insert">
                            Add Doctor
                        </button>
                    </div>
```

10 Session Management

```
12 response.setHeader("Cache-Control","no-cache,no-store,must_revalidate");
13
14 if(session.getAttribute("userid")==null){
15     response.sendRedirect("adminLogin.html");
16 }
17 %>
```

Σε όλες τις σελίδες jsp οι οποίες αφορούν ένα συγκεκριμένο χρήστη είναι αναγκαίο να ελέγχουμε αν έχει δημιουργηθεί το session (αν έχει γίνει login) διαφορετικά γίνεται ανακατεύθυνση στην αντίστοιχη σελίδα του login του κάθε χρήστη. Κάθε φορά που γίνεται login ξεκινάει το session.

```
256  public static void Logout(HttpServletRequest req,HttpServletResponse res) throws ServletException,IOException{
257
258  HttpSession session=req.getSession();
259  session.removeAttribute("username");
260  session.removeAttribute("pass");
261  session.removeAttribute("userid");
262  session.invalidate();
263  }
```

Στη μέθοδο logout αφαιρούμε τις παραμέτρους από το session και κάνουμε invalidate το session.

11 Βιβλιογραφικές Πηγές

Τον κώδικα CSS για τις σελίδες τόν πήραμε από την ιστοσελίδα
https://colorlib.com/wp/template/login-form-v15/
Αλλά και από το bootstrap
https://getbootstrap.com/